Department of Computing Science and Mathematics
University of Stirling

A Peer-to-Peer Network Framework Utilising the Public Mobile Telephone Network

Martin Blunn

Technical Report CSM-190

ISSN 1460-9673

December 2011

Department of Computing Science and Mathematics
University of Stirling

A Peer-to-Peer Network Framework Utilising the Public Mobile Telephone Network

Martin Blunn

Department of Computing Science and Mathematics
University of Stirling
Stirling FK9 4LA, Scotland

Telephone +44 1786 467 421, Facsimile +44 1786 464 551
Email mbl@cs.stir.ac.uk

Technical Report CSM-190

ISSN 1460-9673

December 2011

Abstract

P2P (Peer-to-Peer) technologies are well established and have now become accepted as a mainstream networking approach. However, the explosion of participating users has not been replicated within the mobile networking domain. Until recently the lack of suitable hardware and wireless network infrastructure to support P2P activities was perceived as contributing to the problem. This has changed with ready availability of handsets having ample processing resources utilising an almost ubiquitous mobile telephone network. Coupled with this has been a proliferation of software applications written for the more capable 'smartphone' handsets. P2P systems have not naturally integrated and evolved into the mobile telephone ecosystem in a way that 'client-server' operating techniques have. However as the number of clients for a particular mobile application increase, providing the 'server side' data storage infrastructure becomes more onerous. P2P systems offer mobile telephone applications a way to circumvent this data storage issue by dispersing it across a network of the participating users handsets.

The main goal of this work was to produce a P2P Application Framework that supports developers in creating mobile telephone applications that use distributed storage. Effort was assigned to determining appropriate design requirements for a mobile handset based P2P system. Some of these requirements are related to the limitations of the host hardware, such as power consumption. Others relate to the network upon which the handsets operate, such as connectivity. The thesis reviews current P2P technologies to assess which was viable to form the technology foundations for the framework. The aim was not to re-invent a P2P system design, rather to adopt an existing one for mobile operation. Built upon the foundations of a prototype application, the P2P framework resulting from modifications and enhancements grants access via a simple API (Applications Programmer Interface) to a subset of Nokia 'smartphone' devices. Unhindered operation across all mobile telephone networks is possible through a proprietary application implementing NAT (Network Address Translation) traversal techniques.

Recognising that handsets operate with limited resources, further optimisation of the P2P framework was also investigated. Energy consumption was a parameter chosen for further examination because of its impact on handset participation time.

This work has proven that operating applications in conjunction with a P2P data storage framework, connected via the mobile telephone network, is technically feasible. It also shows that opportunity remains for further research to realise the full potential of this data storage technique.

Acknowledgements

The production of this thesis would not have been possible without the advice, encouragement and support of many people. There are too many to list in entirety, however I would like to give formal thanks and acknowledge the support of the following people:

- **Dr. Mario Kolberg**, University of Stirling
- **Prof. Evan Magill**, University of Stirling
- **Prof. Leslie Smith**, University of Stirling
- **Dr. Michael Wilson**, Formerly of Systems & Networks Ltd
- **Dr. Peter Burtwistle**, Systems & Networks Ltd
- **Mr. Alan Hamilton**, Systems & Networks Ltd
- **Dr. Gerry Black**, East of Scotland Knowledge Transfer Partnership

Thank you to Mario, Evan and Michael for their patience and suggestions as well as technical guidance and tuition throughout the full length of this project. I would not have completed this without their valuable support at various stages.

Thank you to Leslie for the various means of official support I have received from the Department of Computing Science at the University of Stirling, including a contribution towards the final year course fees.

Thank you to Peter and Alan for their advice, encouragement and financial contribution to supporting activities conducted whilst participating in the Knowledge Transfer Partnership. I also acknowledge and thank them for their express permission allowing me to discuss content within this document that was performed whilst working on the joint Knowledge Transfer Partnership programme with The University of Stirling.

Thank you to Gerry for his independent advice and guidance during my time on the Knowledge Transfer Partnership programme.

Chapter 1

Introduction

1.1 Background

Peer to Peer (P2P) networking technologies are considered to be a distributed system and are a part of the established, mainstream network architectures today. A distributed system is defined as 'a collection of independent computers that appears to its users as a single coherent system' [119]. Implied within this statement is that the number of participating computers can vary and all must act autonomously. P2P networks are also described as an 'overlay' functioning on top of other networking technologies and thereby abstracting themselves from the complexities of hardware and operating systems (OS). P2P networking technologies at a fundamental level require three factors to establish themselves effectively:

- **Networked Capability:** A large scale, unified communication mechanism that provide a means to connect nodes participating in the P2P overlay network.
- **Host Technology:** Viable host technology (software and hardware) to host the overlay of distributed node software.
- **Social Adoption:** Wide scale 'social need / desire / acceptance' to use the applications associated with the P2P overlay network.

These factors have come together for P2P overlay networks hosted within the 'traditional' Internet connected, desktop personal computer (PC) environment. The same could now be said for the mobile device environment as the following paragraphs will justify.

Networked Capability: Many factors have contributed to the unification of mobile telephone technologies across the globe, adoption of common standards and increased technological capability being just two of the dominant reasons. The GSM telephone network standard is the most commonly adopted system in the world, with in excess of 4 billion subscribers [79] and more than 650 operators across 220 countries [4]. More significantly however, by 2013 industry analysts Gartner [93] predict that mobile telephones will overtake PC's as the most common device for Internet access worldwide. Therefore the term 'unification' can also be applied to the closer integration of the mobile telephone networks and the Internet.

Host Technology: The use of mobile telephone devices with significant processing capabilities (frequently referred to as 'smartphones') have been around since the early 1990's [58]. Combining the properties of personal organisers and mobile telephones into one 'smart' device seemed a natural evolution of two categories of hardware. From the 1990's onwards manufactures such as Nokia, RIM and Apple have all produced handsets that have increased in complexity and capability enabled by their increase in computational processing power. As hardware evolved to include larger colour displays, keyboards and more recently touch sensitive interfaces, the operating software has also improved considerably. Progressing from basic applications such as a calendar or contact manager, the hosting of fully Internet capable and configurable third party applications on a flexible OS is now commonplace on higher specification devices. In more recent years, industry commentators have made comparisons between smartphones and 'mini computers' as the specifications of the mobile devices continue to improve [58, 164, 9]. More recent examples of these devices include Nokia's 'E' / 'N' series handsets, RIM's Blackberry and Apple's iPhone. All have made significant consumer sales out of an estimated 57.3 million smartphones shipped in Quarter 1, 2010 [117].

Considering the telephone network capability of using a 'smartphone', data rates have increased substantially in the wireless sector since the popular adoption of 2G GPRS / WAP services around 2003/04 [116]. Currently operating at data transfer speeds up to 7.2 MBps, 3G high speed data access to Internet services has allowed the proliferation of data centric applications to appear on mobile telephones. Arguably, users have until recently thought of their phone usage on the Internet as solely through a mobile web browser. However, other data oriented applications such as mapping (e.g. Google's or Nokia's Mobile Maps [10, 12]) or music streaming services (e.g. Last FM's Mobbler [56]) installed onto the device have helped to change users' perceptions.

Social Adoption: Sociological influences are decisive in a different way in guiding whether a new software concept or application is adopted by the public at large. This is particularly applicable to P2P systems where optimum operating efficiency is achieved in large scale implementations; node volumes in the order of thousands up to millions are a realistic expectation [6, 120, 107, 39]. For example Skype, a popular P2P based VOIP application, has online user numbers in the range of millions each day [122]. Although large node volumes are not an essential requirement for operation of a P2P network, it does supports the design objectives of content availability and network reliability. Content of a P2P network, whilst a huge influence on the adoption success (e.g. Napster [65]), is a requirement only for the overlay to the extent that it should be flexible enough to support a variety of shared digital content at the application level. Currently content typically includes text data, pictures, music files and voice conversations but future forecasts include video and television media [72, 95]. Ultimately, as with so many other underlying technologies (like the telephone itself), it could be argued that if P2P works seamlessly for the applications that use it then the user will not actually care about the details of implementation mechanisms.

1.1.1 Consideration of MANETs

Distributed systems can be implemented at numerous conceptual levels by using different structures. In addition to the prior introduced P2P Networks, Mobile Ad-Hoc Networks (MANETs) are another established technology worthy of consideration. A MANET system can be defined as 'a collection of wireless mobile nodes dynamically forming a temporary network without the use of any existing network infrastructure or centralized administration' [50]. The key characteristics of a MANET system comprise: mobile nodes, no centralised network administration, no pre-existing network infrastructure (e.g. 'base stations') and multi-hop communication [26]. MANET inter-node communication is typically wireless and not associated with any one specific protocol. MANETs are typically employed to address network connectivity at the lower levels of the OSI (Open Systems Interconnection) network architecture. BlueTooth, ZigBee and 802.11 WiFi are all examples of protocols that can be utilised within MANETs. Connection is between single nodes with inter-node communication via a 'multi-hop' approach. The dynamic and loosely organised nature of the node interconnection state presents data exchange challenges. A fundamental issue concerning the optimal routing path of data between nodes can arise [59]. Whilst some research effort has been conducted to address this problem [59, 88], some classifications of P2P networks intrinsically accommodate inter-node routing with more determinable results (see section 2.4).

This thesis is concerned with dissemination of data at application level within the OSI model. Gerla et al. [106] describe MANET routing functionality as several OSI layers below that offered by a P2P network and inadequate to provide the services needed by sophisticated mobile applications. Furthermore, for this work node interconnection is across an established mobile and fixed public switched telephone network (PSTN) where the strength of MANETs is not applicable (i.e. communication where there is no infrastructure). With a PSTN as the data transport mechanism, reliable data location and retrieval becomes a higher priority. For these reasons research focused on P2P networking technologies to solve the data distribution challenges. Some research has even been conducted into combining the complimentary properties of these two technologies [106] but is beyond the remit of this thesis.

1.1.2 Capitalisation on Circumstances

A significant proportion of today's Internet traffic is data associated with P2P overlays [169]. Coupled with significant numbers of users online using smartphones (and the prediction of more in the future [43]), a viable ecosystem in which mobile P2P technologies can thrive has been established. Whilst ultimately it is the content and applications sitting on top of the overlay network that will dictate the adoption rate success of the framework, the underlying technology will also have a significant influence. The P2P Application Development Framework (codename Bushfire[1]) discussed within this thesis helps developers overcome the remaining technical challenges and capitalise on these current beneficial circumstances.

Developed as part of a UK Government Sponsored Funding initiative, the enhanced P2P Application Framework was the result of research and development activities conducted between Systems & Networks Ltd. (SysNet) and The University of Stirling. Project funding management was through a scheme called Knowledge Transfer Partnerships based out of the East of Scotland KTP Centre [85]. The project was commissioned to support SysNet's mobile application development commercial activities as well as forge links with local Universities. Consultancy and validation of research activities was conducted through regular liaison with academics at The University of Stirling. Some aspects of the research were demonstrated at the IEEE Consumer Communications & Networking Conference (CCNC) 2009 [7]. The commercial reasons behind the instigation of this project meant that both the timescales and direction of the venture were more guided / restricted than a purely research driven activity. In addition to assembly of the Bushfire Framework itself (see section 4.2), several applications of varying purpose and completeness were also developed and are presented within this thesis (see section 5). The main requirements and limitations for this work, including applicable commercial demands, are shown within chapters 3 and 4, sections 3.1, 3.2.3, 4.1 and 4.2.

1.2 Aims of this Work

The introduction has postulated that conditions are now right for wide scale adoption of P2P technologies within the mobile telephone network environment. However to date the adoption level could be argued as limited, because of several key factors which this thesis proposed to address. Therefore the aims of this thesis are to:

- Provide an appropriate P2P framework that operates on 'smartphone' handsets using an Internet connection across the mobile telephone network.
- Provide an appropriate P2P framework within which third party applications could be easily developed using a clearly defined application programming interface (API).
- Provide observational data, analysis and guidance on operating behaviour both for the implementation discussed within this thesis and for further research effort.

An 'appropriate' P2P framework is defined as a current mainstream technology (i.e. from a desktop computer networked environment) that has properties that are also pertinent within a mobile environment. As chapter 2 will show, to limit the scope of work it was accepted that the most suitable candidate technology would be chosen from a selection of proven technologies already in existence.

1.3 Contributions of this Work

Some authors have indicated that operating a P2P network upon a mobile carrier network is not realistic [130]. However, the material discussed within this thesis offers a practical solution to the current technical limitations that hinder wider scale adoption. Furthermore, a contribution is made to the suggested direction of further research in known areas of weakness and limitation. The most significant contributions of this thesis are:

- Review of existing mainstream P2P technologies to determine the requirements for an overlay appropriate for use within a mobile telephone network environment. (Refer to chapters 2 and 3.)

[1] The name choice inspired by the 'spontaneous' nature with which P2P networks form and disband.

- Enhancement and extension of an existing P2P application towards a P2P framework by devising a clear and uncomplicated API for use by third party applications. (Refer to chapter 4.)
- Devising a proprietary mechanism to navigate limitations introduced by Network Address Translation (NAT) technology within the public data networks. This approach was subsequently integrated with the P2P framework. (Refer to chapters 3 and 4.)
- Development and implementation of three applications using the P2P framework. These were to test and demonstrate its functionality and behaviour. (Refer to chapter 5.)
- Collection of operational data for a small mobile P2P network to assess the methods for optimising operation (such as improving energy and data consumption). (Refer to chapter 6.)

1.4 Structure of the Thesis

This thesis has been structured to present the reader with a logical progression into the Bushfire Framework. The following list introduces the topics covered within each chapter:

- **Chapter 2: Established P2P Technologies.**
  P2P technologies are defined and classifications of the various design implementations are presented. Two of the largest groupings, ’structured’ and ’unstructured’ are discussed in further detail. An overall evaluation of the technologies presented summarises the chapter.
- **Chapter 3: Considering P2P upon a Mobile Telephone Network.**
  Operation of a P2P network within a mobile environment presents challenges that need to be acknowledged within the system design. This chapter introduces the main influences affecting operation within this domain. Discussion also includes Network Address Translation (NAT) as it has a dominant influence on the operation of P2P networks. A chapter summary presents the outcome of relevant design decisions embedded within the BushFire framework.
- **Chapter 4: Development and Implementation of a P2P Framework.**
  Having presented the underlying technology and the operating environment, this chapter discusses the solution that was developed on this research project: the Bushfire Framework. Introduced with the requirements, further design details relating to architecture and implementation of the framework are progressively revealed. Design solutions to the major obstacles of working within the mobile networked environment are presented.
- **Chapter 5: P2P Framework Applications.**
  As part of the research activities, several applications were constructed to work with the Bushfire Framework. The first was intended as a test/development platform whilst the second and third were proof of concept and commercial grade (alpha release) respectively. Working on the principle that developing applications helped to influence design decisions and highlight errors within the framework, this approach yielded good results. The applications also offer some practical context to the hitherto theoretical discussions.
- **Chapter 6: Optimisation of P2P Framework Performance.**
  Once the design of any system has been completed, optimisations are frequently sought to improve performance. The same process was applied to the Bushfire Framework, focusing on a key design parameter/performance metric: energy consumption. Arguments are presented as to why the chosen parameters should be considered as well as reviewing results of experimentation. Conclusions are drawn from the data collated and presented to the reader.
- **Chapter 7: Conclusions and Further Work.**
  The thesis is concluded with a review of the research work producing the Bushfire Framework, covering existing technology, optimisations and suggested further areas of research activity.

Chapter 2

Established P2P Technologies

Adoption of P2P technologies has been rapid and substantial since appearing notably on the public stage in 1999 with the music file sharing service Napster [107]. P2P is a network model that has the concept of equality at the centre of its development philosophy. The operating paradigm is one of mutual cooperation. Implied by this equality is the ability to scale upwards in size. Since passing the 1 billionth internet user in 2005 with an estimated 18% growth rate of new users per year [80], P2P systems offer one of several viable solutions to the escalating problems of distribution and load balancing for data and processing power respectively.

2.1 Defining Peer to Peer

Each node participating within a P2P network is of equal status through equivalence of capabilities, resources and responsibilities [75]. P2P networks can also be described as autonomous in behaviour: they exhibit self-organising, self-optimising, self-protecting and self-healing characteristics [33]. All of these properties are vital for reliable network operation as it is an accepted assumption that nodes will not be available all of the time. A node will join and leave the network at random time intervals and without warning purely because of the different operating constraints for each of the host machines. Each node is capable of communicating with any of the other nodes within the peer network. Communication can be either direct between nodes or via multiple requests to establish the link. Figure 2.1 illustrates simplistically a typical topology of a P2P network. There is a lack of 'centralised server control' for the data communication paths, implying there is a direct interaction between nodes. A single node both utilises and provides services and data to other nodes within the network, mimicking the functionality of both a client and server respectively.

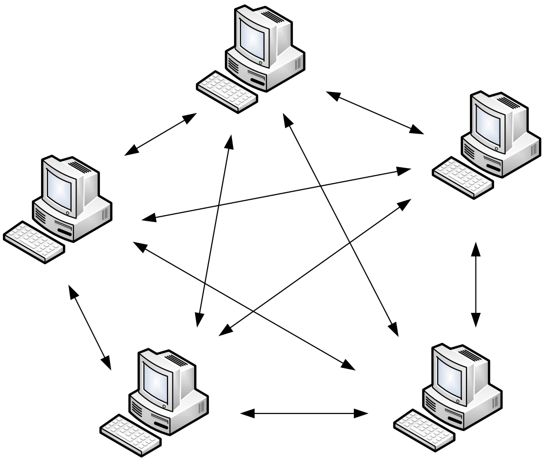


Figure 2.1: Simplified P2P Network Topology

Nodes participating within a P2P network do so primarily to share data amongst each other. The underlying P2P networks put no limitations on the type of data that is shared as this is specified by the application operating at a higher level of abstraction within the P2P system. Whilst practical constraints might exist, any networked application could be linked in this manner. Only issues such as availability of resources will determine if it is the best use of the P2P model in comparison to alternatives such as the client/server model.

Implicit within the process of sharing data storage is the distributed processing and organisation of this data. Constraints also exist relating to connectivity of nodes to the network. Connectivity can be defined in terms of time period or bandwidth and will vary because of the large volume and differing operating constraints of nodes. The frequency of arrival and departure of nodes within a P2P network is referred to as ‘churn’. To compensate for the variable availability of nodes, data resources can be distributed multiple times within a network to ensure the desired level of accessibility. Whilst 100% data availability is only achievable through the use of impracticable levels of resources, good approximations to this are possible by using much fewer resources. Typically, data is also widely distributed across nodes with no reference to physical locality, only network locality.

P2P technologies are referred to as an overlay network. An overlay is the term for a network topology that is constructed on top of another. More precisely, an overlay network is defined as ‘a virtual network of nodes and links built upon the top of an existing network with the purpose to implement a network that is not available upon the existing network’ [65]. In the case of P2P networks, the underlying technology is typically the Internet Protocol (IP) network. This hierarchy of networks is illustrated in Figure 2.2. An overlay network does not take into account physical proximity of nodes, only boundaries in terms of the nodes operating within that network overlay. The various overlay implementations all aim to ensure that each node is equal in line with the criteria of a P2P network. The reality of real world constraints and variations however often means that this is not achieved. For example, the overlay protocol running on each node may be identical but there will be differences in the network connectivity bandwidth, data storage capacity and processing power.

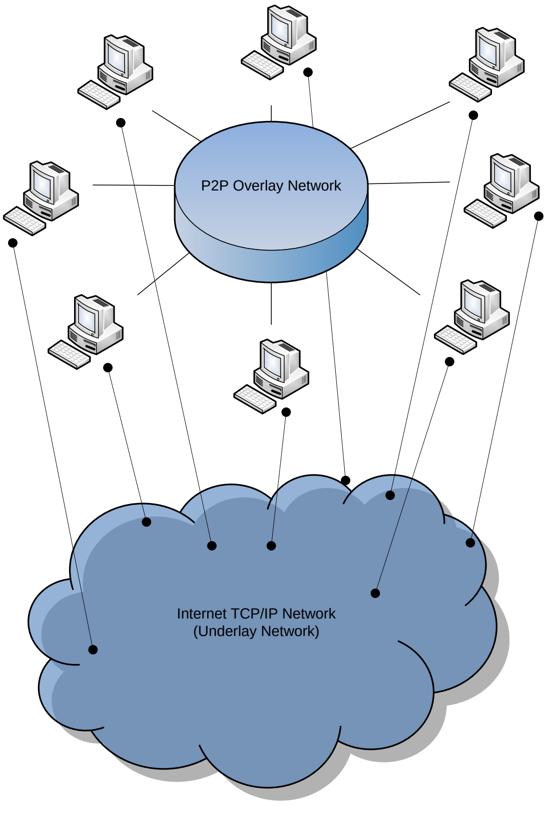


Figure 2.2: The P2P Overlay Network Concept

P2P systems, within the context of the commonly cited four-layer TCP/IP network architecture (excluding the physical layer) [103], belong at the top level 'Application Layer'. Additionally, P2P systems themselves can be considered to be formed of a three layer architecture comprising:

- P2P application layer.
- P2P overlay network layer.
- Underlay network layer (e.g. TCP/IP (Internet) network).

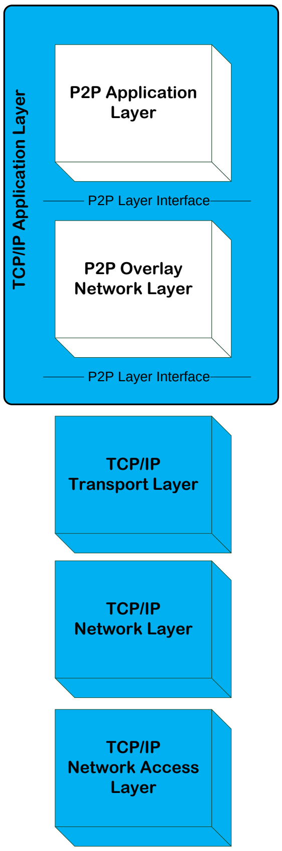


Figure 2.3: A P2P System Layer Diagram

Each layer operates within a set of defined boundaries and communicates across interfaces with the surrounding layers of software. This layered concept is illustrated within Figure 2.3. This design approach ensures scope and design complexity of individual software components is kept within manageable limits. The software infrastructure that implement these algorithms permit the building of more complex services that reside at the application layer such as file sharing and content distribution systems. It should also be noted that P2P system nodes operate 'outside' the view of the Domain Name System (DNS) infrastructure, relying instead upon the search algorithms and identifiers assigned by the P2P network system.

2.2 Peer to Peer Classifications

As the P2P domain has advanced and broadened through academic research and practical application, the different overlay design configurations developed have been grouped into several classifications. Figure 2.4 illustrates the organisation and inter-relation of the P2P overlay network categorisations adapted from proposals by Buford et al [65], Ross et al [107] and Buford & Ross [104]. At the top of the classification hierarchy are two broad groups: structured and unstructured (an approach also supported by Lua et al [62]). This terminology is with reference to the operational approach of the search algorithm employed on each node within the network. As discussed later, a structured overlay exhibits properties of a more organised and efficient network communication system. Below this classification are sub-groups defined by the distinct property of that particular overlay design.

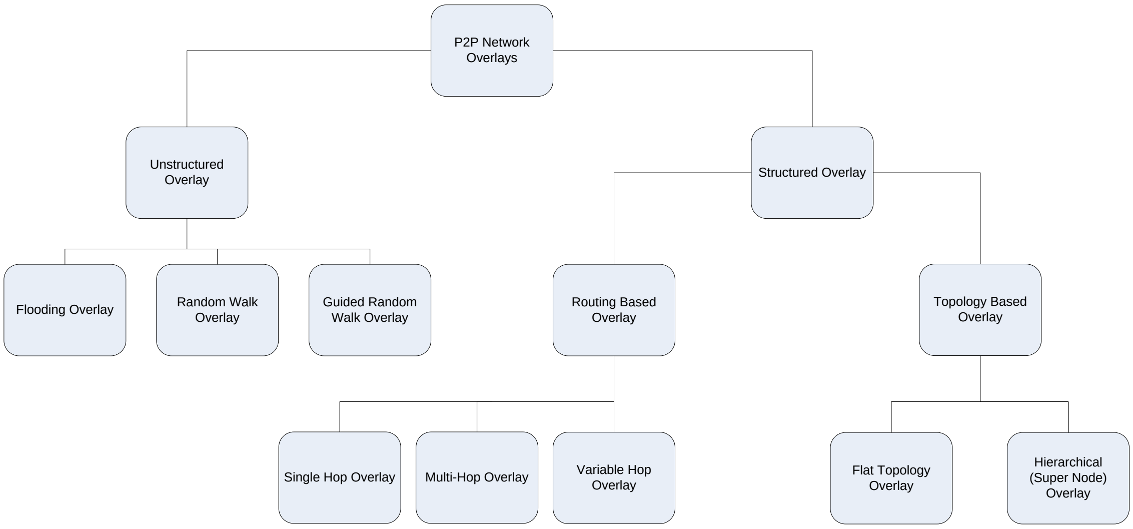


Figure 2.4: Simplified P2P Network Taxonomy Inspired by Existing Categorisation Proposals [65, 107, 104, 62]

From the late 1990s until recently, P2P development and implementation has been primarily within the desktop computer domain using the Internet protocol as the underlying network transport protocol. The growth of Internet activity and access bandwidth from the mid 1990s has fueled the transition of the P2P concept from academia into the mainstream public domain. This transition was instigated primarily by the appearance of music file-sharing web sites such as Napster in 1999 [33]. Since then acceptance of P2P applications has been steadily increasing, with it accounting for around 50% of network traffic through Internet Service Providers in the USA in May 2008 [16]. In order to evaluate the requirements and suitability of a P2P overlay for the mobile environment, an insight is achieved through analysis of the properties of existing categories of P2P overlays.

2.3 Unstructured P2P Overlay Network

An unstructured P2P overlay network is defined as one where the links between the participating nodes are established arbitrarily. Links are established depending upon whether a node has been previously searched or its proximity in the network to the current node issuing the search request. A search query for information in the network is passed throughout the overlay network to the entire node population through a process that appears random. To facilitate the establishment of the links between nodes, an unstructured overlay network can have a coordinating server machine at the core of its network. Designs vary however and the server element is not mandatory. If present, this server would only have the responsibility of coordinating routing link information between nodes. Peers are individually responsible for hosting the data on the network. Even with routing coordination functionality supporting link establishment, unstructured overlay networks permit varying and unknown levels of data repetition within the network. Unstructured P2P overlay networks have both advantages and disadvantages as a mechanism for storing and retrieving data amongst network nodes. For this design approach they include:

- *Advantage:* Relatively simple implementation computationally, compared with a structured design approach, reduces complexity of software operating on peer nodes.

- *Advantage:* Low overheads in terms of processing or communication requirements for a new peer to join an existing network. A node typically copies routing information from another peer present on the network.

- *Disadvantage:* Search query broadcasting causes a high amount of signalling traffic within the network, which directly competes for bandwidth along with the desired data.

- *Disadvantage:* No correlation between the peer and the content managed by it.

- *Disadvantage:* Search queries for data may not always be successfully resolved as normally this type of network has poor search efficiency. The design does permit a situation where a query across a network will not find a peer with the desired data even when it is present.

The unstructured overlay implementation was common in designs during the early years of public adoption of P2P networks. This was primarily due to its relative ease of implementation and limited impact on processing resources of the node host machine. The simplicity of design promoted flexibility and adaptability which helped the widespread adoption of P2P systems in general. More recently the focus of academic activity has switched to structured overlays as the cost of processing power and data storage has diminished. However, unstructured overlays remain present in the public domain, implemented in some commercial systems such as Gnutella [143], eDonkey [141] and BitTorrent [139].

There are two sub-categories of unstructured P2P overlays that reflect the general routing behavior of the nodes within their network. The efficiency of either system in comparison to structured overlay networks is comparatively low [65]. However, when used on hardware and IP networks that have generous resource allocations of communication bandwidth and/or storage capacity, these overlay designs provide a perfectly viable mechanism for resource sharing. The two categories of overlay presented are termed ‘Flooding’ and ‘Random Walk’ which trade off speed and efficiency against each other.

2.3.1 Overlay Classification: Flooding

Network overlays fitting this classification exhibit behavior where a piece of data is retrieved from the network by placing lots of query requests to all of the participating network nodes across the entire network. This action is referred to as the process of ‘flooding’ the network with data requests. This approach has little coordination between nodes and requires only relatively simple processing algorithms to be implemented within each node. This algorithm primarily implements a ‘familiarity forward blocking’ logic to prevent search requests that have been seen on a particular node re-propagating from it. However the trade-off for a simple implementation algorithm is the inefficient propagation of a large number of request messages around the network, usually resulting in either the extreme situations of no data or multiple data sources being found. The flooding approach, whilst not particularly efficient, is fast compared with the alternative search request method employed in the ‘random walk’ overlay network. This flooding approach was adopted by many of the early successful public file sharing networks such as Napster and the later developed open-source Gnutella. These implementations are frequently described as disruptive technologies as they proved the concept that truly wide scale operation of P2P overlay networks was possible. These particular implementations are no longer used in great numbers but a modern successor that exhibits ‘flooding like’ behaviour, Bit Torrent, continues to remain popular. Bit Torrent has found a use as one of the most common protocols for transferring large files across the internet. One internet usage report indicated that in 2008/09, Bit Torrent accounted for approximately 27-55% of all Internet traffic depending on geographical location [34].

2.3.2 Overlay Classification: Random Walk

The ‘Random Walk’ algorithm is considered to be a primitive but justifiable search mechanism within unstructured overlay networks [108]. It represents a classification within P2P overlay networks where data requests are made with what appears to be random nodes that are dispersed across the network. Search requests are simply propagated from one node to an adjoining node repetitively until the desired data is located or the search is stopped. The difference between this approach and the flooding technique is that dispersion of search request messages is more controlled and thus makes more efficient use of the network bandwidth. The node search algorithm primarily consists of logic implementing a ‘hop’ or ‘Time to Live’ (TTL) counter to prevent searches propagating indefinitely around the network. Efficiency is improved further when the overlay network is in one of two circumstances: highly clustered or minimal node churn [108]. The negative impact of this approach however is the time taken to find the desired data within the network. Given that request dissemination is controlled and finding the data is based upon random selection of the correct node, time taken to retrieve the data can vary but will usually be slower than the ‘Flooding’ technique. This network overlay approach was adopted by academics in early theoretical studies [65, 108, 168] of P2P overlay networks but appears to have been eclipsed by the popular aforementioned flooding based overlays such as Gnutella and Bit Torrent.

2.3.3 Overlay Classification: Guided Random Walk

The third classification is primarily an extension of the Random Walk algorithm designed to add organisation to the task of navigation through the node search space. Organisation is typically achieved through keeping information on the surrounding nodes of a particular node within the network. Information may consist of either probability samples of the likelihood of data being held on a particular adjacent node [113] or 'traces' of the content itself that allude to its existence [90]. The resultant outcome of these approaches, in comparison to the other unstructured classifications, is a reduced level of network traffic and shorter search times when requesting content [90]. Several academic implementations of network overlays within this classification exist [113, 90, 112] but it largely remains outside the commercial domain at present as focus has moved towards implementations of structured overlay networks.

2.4 Structured P2P Overlay Network

A structured P2P overlay network requires nodes to be logically organised with respect to each other in the system. Nodes are commonly organised in geometric structures that promote inter-connectivity and routing efficiency. The coordination of nodes in a logically organised manner is achieved by applying an identical routing protocol across all nodes participating within the overlay network. The result ensures both communication messages and the data storage/retrieval process occurs in a controlled, efficient manner. These efficiencies are gained in both bandwidth and storage capacity reduction through the minimisation of inter-node communication and unnecessary data replication respectively.

Structured overlay networks exhibit the common property of 'key based routing' where a unique identifier is assigned to the peers forming the network as well as the data objects being stored within it. Both unique identifiers are from within the same address space, so the overlay algorithms associate object data with the closest matching peer identifier as a means of organising the data. The P2P systems implementing a structured approach are also referred to as 'DOLR' (Distributed Object Location and Routing) systems. These represent the most popular category of P2P networks today and within this is a major sub-category referred to as 'Distributed Hash Tables' (DHT) [65].

DHTs are intrinsic to the operation of many of the principal structured P2P overlay algorithms. Data that is stored using this mechanism is assigned to nodes that are dispersed across the network using mathematical hashing algorithms (designed to attempt a statistically even allocation spread). The hashing algorithms are used to generate the keys described above in a consistent, reproducible manner. As each peer is responsible for a range of data object keys, the result of the hashing algorithm ensures that network traffic and commitment of resources is dispersed across all nodes. The structure of a DHT P2P based system integrates conceptually into a three layer hierarchy as shown in Figure 2.3 where the DHT functionality is encapsulated within the P2P overlay network layer. The general properties of a structured P2P overlay network include:

- Peers within a network act as both client and server of both data and routing information for other nodes participating in the network.
- There is no central server or router managing the network; peers explicitly take on this responsibility through a coordinated effort of communication between themselves.
- When Distributed Hash Tables are used, it ensures every peer and its content is tagged and traceable across the network.

The advantages and disadvantages of structured P2P overlay networks include:

- *Advantage:* Search efficiencies are greater than the unstructured approach due to the structured (repeatable) protocol used to determine peers responsible for the data.

- *Advantage:* Signalling traffic is reduced in comparison to unstructured network as the search criterion uses a known protocol optimized towards reducing the search time and effort.
- *Disadvantage:* Increased complexity of the routing/communication software operating upon each of the nodes within the overlay network. The efficient use of network bandwidth and storage resources has been traded off against implementation complexity and processing power requirements.

Structured P2P overlay networks can be broadly categorised from either of two design perspectives: routing or topology. The routing classification concentrates on the method by which user and network data is transferred and stored within the network. The topology classification considers the logical organisational arrangement of the nodes within the network. A specific structured P2P algorithm could be associated with both of these categories as it will exhibit properties that are applicable to both. However the two categorisations provide a reference framework through which to consider the numerous overlay algorithms.

2.4.1 Distributed Hash Tables

Distributed Hash Tables (DHTs) are a fundamental core element of structured P2P network overlay designs. These are a class of decentralised distributed systems that provide a lookup service similar to a hash function lookup table. DHT based systems exhibit the following general properties:

- Decentralisation of resources incorporated into design.
- Scalable in operation.
- Tolerant of faults or harsh operating scenarios.

In essence, data storage is allocated to a specific node within a DHT P2P overlay network as a (key, value) pair using a consistent repeatable process: a hashing algorithm. The creation of a (key, value) pair is determined firstly by operating the hashing algorithm upon the value (i.e. data to be stored) to form an (almost) unique reference tag for it referred to as the key. Additionally each node forming the peer network is allocated a unique identifier and is hashed with the same algorithm so that it is part of the same 'numerical range' as the (key, value) pairs associated with the data. Then the protocol running on each node allocates the (key, value) pair to the storage available in the peer node with the closest matching hashed identifier to the data key. Fundamental to this is that all nodes contain the processing algorithm to match the key and data; therefore any participating node can efficiently retrieve the value associated with a given name. Figure 2.5 illustrates (using hotel addresses as an example) the process of hash tagging data with a key of a format suitable for storage within a DHT based distributed network (hash tag identified nodes). The process is shown with abstract values for the Keys (K1, K2, etc.) and Node identifiers (N1, N2 etc.). In reality, the keys and identifiers are long sequences of alphanumeric characters generated by processing the 'clear values' with a Hash function (e.g. the result of 'hashing' the Royal Hotel Address is Key K5). Furthermore, importantly both the values are still within the same numeric range and can be processed for similarity / proximity via a suitable algorithm. The illustration shows a simple situation where search term keys are stored upon the node with the matching node identifier. Note that a node does not have to have an identifier that matches a key exactly in order to have 'key/value' pairs associated with it; pairs are assigned to the node ID that is numerically closest to the key. Given the very large range of values that could be generated from a hashing function; the likelihood of exact matches between data key and node identifier is very slim. More typically data keys are placed upon the node with the closest matching identifier. Different implementations have different algorithms for determining this closeness and assigning a suitable node. This approach can vary amongst implementation designs, but importantly all nodes participating in the P2P network will be using the same algorithm to produce consistent results.

The key to the successful use of DHTs within this type of overlay network is that the operating protocol on each participating node actively maintains this mapping of node identifiers to the data (name, value) pairs throughout the dynamic lifetime of the network. Therefore, with the responsibility for 'node to data' mapping devolved amongst all the nodes within the network, changes such as the variation of participating peers causes minimal disruption to data retrieval. P2P networks by their very nature are fluid in construction. This DHT based mechanism permits very large scale implementations as well as handling the continual arrival, departure and failure of nodes. In essence, the properties of DHTs permit the (almost) continual availability of data in a network of nodes where there is no control over connectivity.

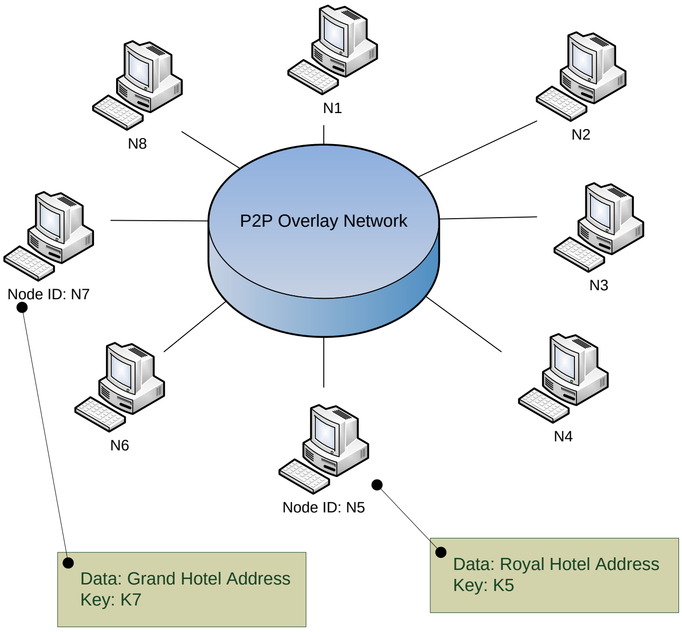


Figure 2.5: Simplistic Overview of Distributed Hash Table Operation in a P2P System (e.g. to Store Hotel Address Information)

As with any technology, DHT based P2P systems have both advantages and drawbacks. The process of allocating a hashed key to a piece of data helps with the even spread of resources across the participating nodes in the network. DHT hash algorithms exhibit a mathematical property for even dispersal of resultant hash codes that are both repeatable and (almost) unique. For example in Figure 2.5, the search term (or data) 'Grand Hotel' is similar to 'Royal Hotel' in pure language terms but from a hash function perspective the resultant hash key (alphanumeric sequence) is very different. Hence, the data associated with either of these hotels is most likely to be stored upon different nodes (assuming the network is large). However the properties of the algorithm that allow the generation of a wide range of hash code sequences also become the drawback of this approach. The precise nature of a hash key means that any difference in the term that is processed by the hash algorithm (no matter how minor) will yield a very different hash key. In the context of a distributed P2P system, where a search for data is performed using the key associated with data, there is the implication that the search term must be exactly the same as the term used to tag the data within the network. Any variances in the search term will not return the desired data. Thus in Figure 2.5, where the example data has been 'tagged' with a derived hash code of 'Grand Hotel', a search for data using the term 'The Grand Hotel' will not be successful as a different key has been generated. This limitation of using a precise algorithm in a keyword searching system, where a certain level of ambiguity is inherent within it, can be minimised by the use of multiple search keywords tagged to the same data. In general, structured systems using DHTs at the core of their design benefit from the simplicity and consistency of creating unique data references and sharing these amongst participating nodes.

2.4.2 Structured P2P Overlay Networks: Routing

Considering overlay networks from a routing perspective produces three distinct groupings. These groups relate to the operation of the search algorithm employed to find user data within the network. Searching within a structured overlay network is related to how quickly and efficiently results are returned to the querying node. These criteria correlate to how much communication bandwidth burden is placed upon the P2P network. In essence, there are two groups of data within a P2P overlay network: one group associated with the user data and the other associated with maintaining the network itself. The network data is frequently referred to as routing data as it configures the organisational layout of the overlay. The design of the routing behaviour has implications on the number of queries of other network nodes that need to be made before the correct user data is found. The number of queries is referred to as the 'hop count' and is related to complexity and responsiveness of the network. Each hop will either return the user data itself to the requesting node or information on where to continue the search. The concept of the 'hop' being the routing path through an overlay network is more easily visualised through Figure 2.6 and Figure 2.7. There are three classifications for the number of routing hops employed by the search algorithm within an overlay network: single, multiple and variable. Variable Hop searching is a conceptual variation of the multiple routing hop approach. As their names suggest the classifications refer to the number of peers contacted to convey the routing message from the source node to the destination node. Typically a hop count can refer to either an average count or a 'worst case scenario' which relates directly to the time taken to find the requested data. The idea of routing hops is most clearly illustrated with the conceptual diagram used to explain the logical organisation of some structured P2P networks. Structured P2P networks are often represented as ordered rings of nodes where the sequence is the numerical identifiers assigned to the nodes by the routing algorithm. For simplicity these identifiers are illustrated unhashed as 'Node N' numbers shown in Figure 2.6 and Figure 2.7. In reality, as previously indicated, these identifiers are actually hashed with the same algorithm that is applied to the data to produce a key for the data.

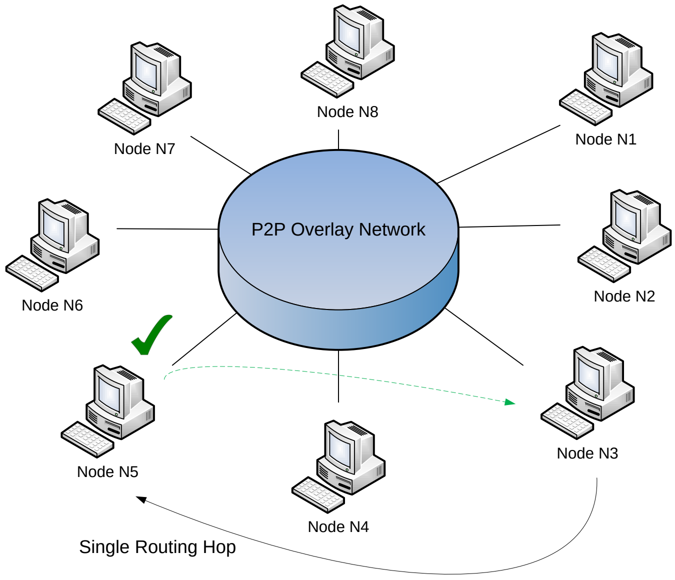


Figure 2.6: Single Routing Hop Concept within a Structured P2P Overlay Network

Figure 2.6 and Figure 2.7 illustrate that searching for data within a network can imply communication either between just the 'searcher' and 'holder' of the desired data or can include other intermediate nodes. The following sections discuss the properties of the structured overlays for each of the hop classifications with descriptions of example implementations.

Routing Overlay Classification: Single Hop

Single Hop overlays permit the data requested from the network to be found in a single network call. Each node possesses a full routing table of all the other nodes in the P2P network. This routing table contains information about each node's corresponding underlying IP address and the data that is stored on it. Single Hop overlay networks have the advantage that fast data search and retrieval times are possible and less network traffic is generated in the request phase in comparison to multi-hop systems. However drawbacks of this strategy include the assumption that the node containing the desired search data is held within the routing table. Additionally, updates are required to keep the routing table data fresh each time a node joins or leaves the network. Each of these updates introduces network traffic and so the churn rate (or frequency of nodes joining / leaving the network) can impact significantly on the volume of data transferred. The operational scenario of the P2P network can have a large impact on the rate of churn of its participating nodes. The scenario in this case explicitly refers to spare storage and processing resources as well as internet connectivity / bandwidth. For example a network of office

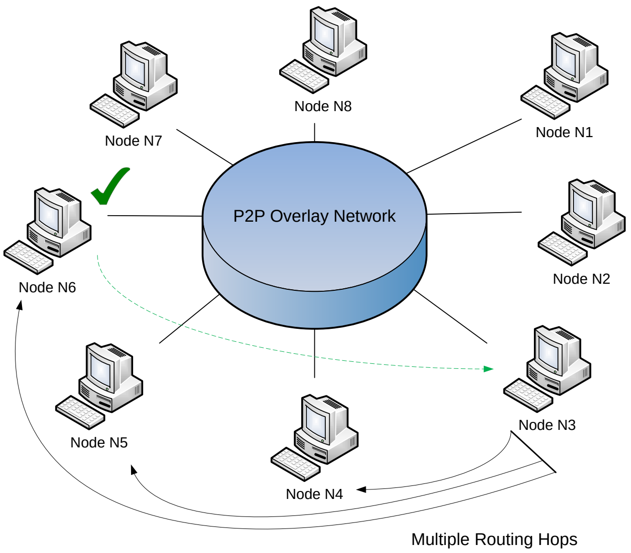


Figure 2.7: Multiple Routing Hop Concept within a Structured P2P Overlay Network

desktop computers is more likely to be a stable environment than laptop devices belonging to a traveling sales force. Furthermore, the processing and storage of routing table data for the entire P2P network is trivial for small numbers of nodes. However as numbers increase into the millions (which is a commercial scale expectation) the resource commitment is more substantial. All of these factors were envisaged as inhibitors to the adoption of Single Hop systems when research commenced on this aspect of P2P systems. However, availability of computing resources has increased significantly over the intervening time period and has alleviated this problem. In essence, users now believe that computing processing power, storage and data bandwidth allocations available to them are either sufficient to meet their current needs and/or are worthwhile committing for the returned benefits. Benefits may include access to digital content, given that one of the most popular uses of P2P systems since their creation has been file sharing [15, 73, 89].

An example implementation of this overlay classification is called EpiChord. EpiChord evolved from a multi-hop system called Chord as a result of attempts to reduce the number of network calls made when searching for data. In the EpiChord system nodes are organised in a hierarchically flat, circular structure and allow the distributed storage of values across all participating nodes. This is achieved through using the previously discussed DHT technique with a consistent hashing mechanism to create keys and node IDs. Each node has a routing table (also known as a finger table) within it to allow a request for a certain key to be routed to the node that holds the ‘key/value’ pair within the network. As discussed earlier, this approach assumes that each table is kept up to date with valid data. With EpiChord this is achieved through a node actively monitoring network traffic that is passed around the overlay and updating its routing table entries as appropriate. This activity is supplemented by dispatching a specific routing table ‘refresh request’ message. These messages or probes of the network are only initiated as a backup mechanism if lookup traffic levels are too low to support the desired level of network performance. Epichord’s reactive state maintenance strategy does not however keep the routing state as up-to-date as a proactive strategy.

Routing Overlay Classification: Multi-Hop

Multi-Hop overlays permit the data requested from the network to be found in a varying number of network calls dependent upon specific network configuration factors. Factors will include the specific node making the search request, the proximity (in terms of routing) between the search node and target and the ‘freshness’ of the node routing table. These factors are present because each node possesses a routing table that contains data representative of only a selection of other nodes within the P2P network. The routing data granularity varies with proximity of the predicted node hosting the required data to the node performing the search request. A routing table will have numerous table entries on nodes within close proximity but only a sparse amount of information on nodes further away. In essence, a node knows most about its immediate neighbours in the logical address space, dwindling for more distant nodes. This segmented arrangement of routing data within the network ensures that:

- routing table sizes do not become too large to operate adequately - even when connected node numbers are millions in magnitude.
- data will always be found on a network if it is present as nodes will either know of the data directly or will pass on information about a node that could know.

Multi-hop overlay networks present more variable system behaviour than single hop systems in terms of content search times and network data bandwidth. If the host node of the desired search content is known to the requesting node, response time is quick and a minimal number of network calls is placed on the network to retrieve the content. However, if search requests are dispatched to nodes beyond those recorded in the requester's routing table then extended search times are incurred and greater network bandwidth is consumed. Node churn also has an impact on these figures as a higher node churn rate will mean that routing table data will become invalid more quickly, resulting in search requests being sent to invalid nodes. However, conversely, multi-hop systems are best suited to finding data in a network when there is a high churn rate as the core algorithm behaviour ensures that a search of successive nodes will continue until either the desired content has been found or all of the eligible nodes have been inspected. Another approach incorporated into multi-hop systems to handle node churn is parallel data requests. As implied, multiple search requests for the same data are issued in order to determine a result more quickly. As an indirect consequence, this approach also disperses the processing load placed on individual nodes across the network by minimising excessive demands upon a few 'popular' nodes. Examples of overlays in this category include Chord, Pastry, Tapestry, CAN (Content Addressable Network) and Kademlia. These examples are amongst the most popular of overlay designs available but do not represent an exhaustive list. All of these technologies listed, with the exception of CAN, work in a similar manner and have their participating peers arranged in a logical circular ring. CAN as an alternative works by organising data across a multi-dimensional cartesian coordinate space but has the same aims of being distributed, scalable and fault tolerant [115].

For the logical ring methodology, as with the single hop structure, routing table information correlates distributed keys derived from the unique node identifier and the content stored within the network. Keys are again subject to a hashing algorithm to ensure even distribution around logical overlay ring. Kademlia is an example of an overlay network that achieves its routing organisation through a simple algorithm involving XOR logic processing on the hashed node identifier or content. Whilst control of the routing table maintenance is autonomous from the user in Kademlia (as in all P2P overlays), there is a direct link between commands issued at application level and those processed within the overlay network. For example, in the Kademlia overlay, a generic task such as content storage or retrieval, is broken down into one of four distinct basic operations: 'ping', 'find_node', 'find_value' and 'store'. Each node will use one or a combination of these operations to complete the assigned task. Therefore routing table updates will make use of the ping command to check a node's presence whilst storing data will use find_node and store operations [148, 65].

Routing Overlay Classification: Variable Hop

Variable Hop overlays are the latest in the evolutionary line of overlay designs extending efficiencies by combining the beneficial aspects of both single and multi-hop topologies. The overlay will vary the network bandwidth it consumes in maintaining the routing data to compensate for the inefficiency that churn brings to a network. Bandwidth consumption will directly depend upon the size of the routing table and the hop strategy employed at that point in time. As the size of a network expands and contracts, varying the approach to recording the routing data of the nodes present can ensure low latency in fulfilling search requests. Variable hop overlay designs go further in optimising routing maintenance efficiency by allowing for connection bandwidths available to individual nodes [65]; in essence more routing maintenance activity is assigned to those nodes with greater bandwidth allocation. This performance characteristic of variable hop overlays aligns itself closely with requirements for operating in a mobile domain, where nodes may have differing data bandwidths available and a differing times dependant upon their local mobile network connectivity arrangements. For example, urban, densely populated areas have a higher density of mobile network cells than rural regions and are therefore likely to offer a mobile user greater connectivity bandwidth due to a more favourable contention ratio, proximity and power configuration. (Note this is not exclusively the case as low data rates can be experienced at locations where there is a high user concentration, such as railway stations, putting excessive demand on a single network access point.)

Accordion, is an example implementation of this overlay classification which has the objective of reducing routing latency by adapting a node's routing table size dependent upon its available connection bandwidth and churn rate. Accordion, as with other overlay designs, uses consistent hashing techniques to assign keys to nodes arranged within a logical circular identifier space. Unlike the variable hop overlays, however, Accordion uses recursive (rather than iterative) parallel node lookups to update the state of the routing table on a particular node. The node requesting a lookup from a peer selects a destination based on not only the key it has present in its routing table but also in relation to applicable gaps in the routing table [65]. In essence, each node learns of its neighbours through a combination of deterministic and opportunistic routing algorithm decisions. This approach also helps the node to maintain a tradeoff within its routing table configuration between an accurate table (and thus improved routing performance) versus high bandwidth consumption [60]. As illustrated by the performance graphs of Buford et al. [65], lookup latency is improved significantly with Accordion's parallel lookup approach compared with the other P2P topologies, but at the expense of high network bandwidth consumption. To allow for this, the number of parallel lookups is adjusted dynamically in proportion to a node's capability to handle network traffic and the level of traffic at that time.

2.4.3 Structured P2P Overlay Networks: Topology

Understanding a structured P2P network overlay design from a topology perspective requires consideration of the logical layout of the routing network rather than its implementation mechanism. Designs are described in their conceptual or schematic state and are categorised into two broad groups: flat and the less common hierarchical category. These groups also infer the status or hierarchy relationships that exist between nodes.

Structured P2P Overlay - Topology Classification: Flat

The terminology 'flat' in this case refers to the logical hierarchy of the nodes within the network, each peer being equal and accessible to all others. The hierarchy is organised around the classification of the node identifiers. When new nodes arrive or depart the network they are required to fit within the logical identifier space created by the other nodes already present on the network. This occurs because of the uniqueness and distributive properties of the hashing algorithm as described within section 2.4.1. The overlay technologies described in the single and multi-hop section, such as Epichord and Kademlia, can also be classified as flat in design. Figure 2.8 illustrates the logical arrangement of a flat topology network that is typically considered as a logical ring of nodes.

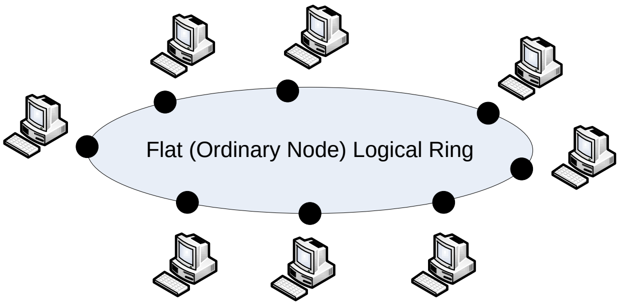


Figure 2.8: Logical Ring of Nodes illustrating a Flat Topology Overlay Classification

Structured P2P Overlay - Topology Classification: Hierarchical

Hierarchical or 'super node' design overlays are a hierarchical arrangement of nodes which have differing capabilities, priorities and status within the network. Super nodes can be any node within a P2P network that takes on additional duties associated with network relaying or acting as a proxy server. Depending upon the specific design of the overlay implementation, assignment of a node to 'super node' status can either be user defined or happen automatically. Although not mandatory, super nodes can have increased resources available to them to cope with the additional load compared to 'ordinary' nodes. Extra resources typically include greater CPU processing capability or a larger connectivity bandwidth. Additionally, super nodes are frequently selected with reference to their availability within the overlay network given the increased traffic that these nodes route. Communication within a super node topology network occurs between nodes at either the super node or ordinary node level as well as between the two. A super node will host a routing processor that will be compatible with the routing algorithm of the 'ordinary nodes'. Additionally it will include logic restricted to routing data exclusively between the super nodes. As illustrated in Figure 2.9, super node networks are considered as two logical rings of peers, one with a differing status from the other.

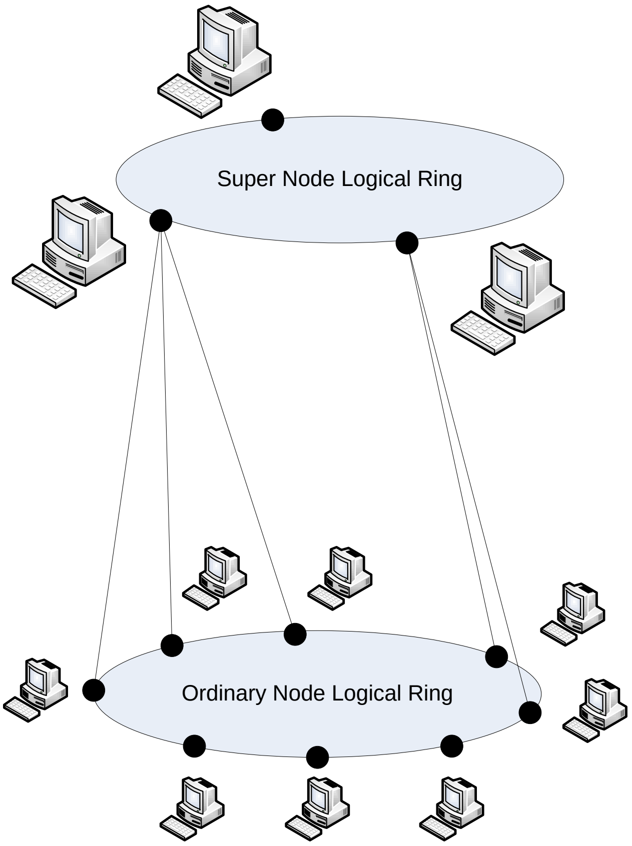


Figure 2.9: Hierarchical (Super Node) Topology Overlay Classification

This two tier, vertical ring structure promotes segmentation and localisation of both data and communication flow within the network. Segmentation using a super node structure is a useful property to promote within a P2P overlay network for several reasons:

- when 'ordinary node' numbers are significantly large (order of millions) which promotes manageable scalability.
- distribution of nodes is logically grouped by some influencing factor such as network or physical locality (e.g. nodes located within a particular country) which can promote lower latency response times.

- when routing of data must return a reliable availability result (e.g. a routing path must always exist between nodes wherever they are present on the network)
- when communication between nodes must meet fixed latency times (e.g. routing to meet high ‘quality of service’ agreement levels for voice or video)

Super node based overlay architectures have been present since 2003 through the use of the Freenet protocol present in early systems such as Kazaa [149] and Grokster [144]. Another example of an implementation technology that uses this design was an unstructured P2P overlay called FastTrack [142]. Experimental work conducted with FastTrack in 2003 gives a representative illustration of the proportion of nodes under the two classifications in this type of network: for approximately 3 million nodes in the overlay, 25,000 to 40,000 were classed as super nodes [65].

The concept of super nodes still exists within current popular technologies such as the proprietary VOIP application Skype [123]. The Skype P2P communications network can have up to 20 million users (or nodes) online across the world at peak times [122]. The super node structure is used to record and convey network presence information (i.e. user online status) but not in the routing of calls between users [49].

2.5 Evaluation of P2P Overlay Networks

This chapter has presented information on several classifications of P2P overlay; however some are more suitable for application to a mobile domain than others. Consideration has focused upon the routing overlay classifications primarily because this will have a greater impact on the performance of the overlay network from the application perspective. Section 2.4.3 has shown that there are some advantages to running a hierarchical network instead of a flat topology overlay but these should be considered against implementation and processing complexity. Additionally scalability issues, a principal justification for using hierarchical overlays, are not considered an immediate problem as flat topology networks will still handle node numbers of magnitudes in the tens of thousands with acceptable boot/network join times [19]. Scalability issues in terms of routing efficiency also steer considerations away from unstructured implementations and towards structured overlays. As the majority of structured routing based topologies are also flat in design this guides the decision process to choose a suitable mechanism from the three routing classifications discussed: single hop, multi-hop and variable hop. Whilst the long term goals of variable hop systems might closely align with the unique requirements of a mobile system, such as limited data connection bandwidth, this is still a relatively immature technology compared with single and multi-hop systems. However, the ability for a variable hop system to accommodate low bandwidth nodes whilst minimising any adverse impact on the performance of high bandwidth nodes could become increasingly important; the difference between slowest and fastest internet connection will only continue to get larger. Single hop overlay systems have been shown to work best when node churn levels are low in order to prevent stale routing table data or excessive numbers of routing maintenance messages. However, given the operational scenario of a P2P node hosted upon a mobile device, frequent churn is a real possibility. For example disconnection from the overlay network could occur due to factors such as network coverage, battery life or data channel termination due to voice calls. Additionally, there is also a processing overhead associated with hosting and maintaining a routing table representing a large overlay network. Unlike overlay nodes hosted on desktop computers, with mobile devices it is clear this processing represents a larger commitment of limited handset resources (although exact impact will vary across host hardware). Case studies on an unstructured overlay called Gia [65] have also indicated an efficiency weakness termed a collapse point that is related to the number of queries in a network for a given network size and peer capacity [65]. In essence, it has been shown that increasing the number of queries in the network results in a saturation point beyond which performance degrades. To avoid this scenario, networks are preferably configured so that they have a large collapse point. This typically translates to wasted peer capacity for a network as the average number of nodes does not require the same resource commitment. This inefficiency, whilst tolerable on desktop computer hosts due to generous resource allocation, is not ideal within the resource-limited handset environment.

Therefore, multi-hop overlay systems present themselves as the most suitable overlay classification to be considered for adaptation in the mobile domain. Properties such as optimised routing tables (i.e. detailed information for nodes closer in overlay proximity), parallel query ‘lookups’ and iterative search process (i.e. versus recursive approach in single hop systems) all serve to reduce lookup latency and aid avoidance of lookup timeouts. Parallel address ‘lookups’ also reduce the effect that outdated routing tables can have on lookup latency by regularly updating routing information with monitored data from search queries processed. This maintenance strategy allows large amounts of routing state to be maintained with only a modest amount of bandwidth. The iterative search process compensates for the high node churn rate potentially encountered on a mobile overlay by using an exhaustive search strategy where all valid nodes for hosting the data have been queried. All of these characteristics contribute to an overlay design that is considered scalable and robust [65].

Chapter 3

Considering P2P upon a Mobile Telephone Network

When considering the requirements for a mobile telephone network P2P overlay, it is necessary to consider them from two perspectives: requirements of any software operating in a mobile environment and those specific to P2P systems. As visualised in Figure 3.1, requirements are ‘pushed down’ to the overlay from the applications that run upon it and dictated by the network environment upon which the overlay operates. The influencing factors (technological, commercial and sociological) are common to all three layers in this figure although the level and impact of the influence will vary according to specific criteria. Furthermore, the requirements placed on a mobile P2P system by the hardware technology it utilises oversee the specification of any solution. These factors however are beyond the control of most application developers and so ultimately form an absolute boundary within which an optimised P2P system can be developed.

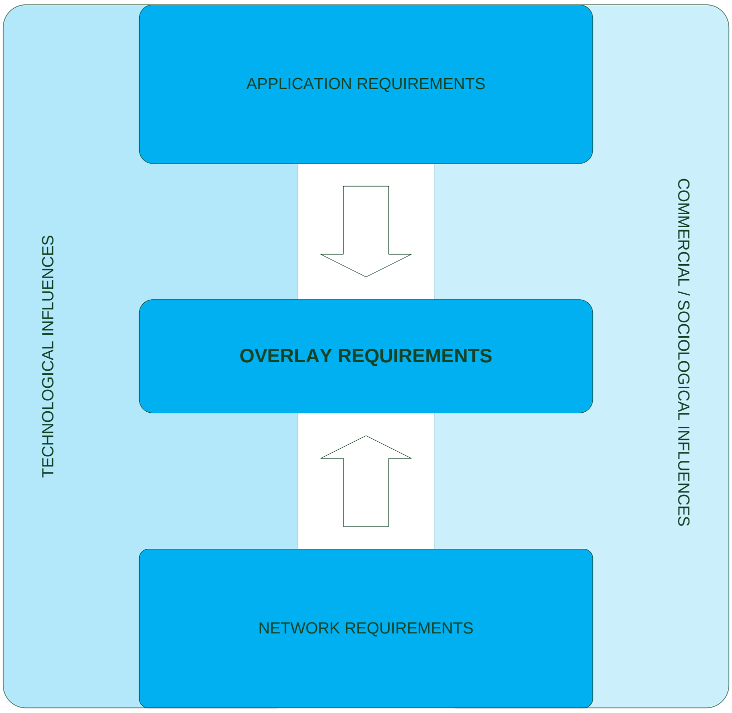


Figure 3.1: Requirements Composition for a Mobile P2P Overlay

Due to the funding mechanism for this research project, as introduced within section 1.1, commercial influencing factors were significant yet typical for an economically sensitive development task. A short development time frame and limited resources (both human and hardware) bound the total development effort to 24 man-months including all supportive activities (e.g. educational and test activities). A restricted choice for target hardware, thereby limiting the software platform selection, also heavily influenced the direction taken by the research. In the case of this project, the target telephone hardware was selected by the commercial sponsor as the Nokia Symbian Series 60 (S60) smartphone platform [158]. A decision driven by internal availability of skills expertise and the appropriate development tool suite, this platform had been adopted by the commercial sponsor for reasons of public popularity. By intrinsic association, selecting the Nokia smartphone platform implied an operating system choice of Symbian [161] and a primary development language of C++. Implications of these technology decisions are revealed within section 3.2 and chapter 4. This chapter covers the application of P2P to a mobile domain through three broad sections, each discussing the salient points of the topic:

i Influences Affecting Development of a Mobile P2P Overlay (Section 3.1).

ii Choosing a Mobile P2P Overlay (Section 3.2).

iii Working Within Network Constraints: NAT Traversal (Section 3.3).

3.1 Influences Affecting Development of a Mobile P2P Overlay

As discussed previously, nodes participating in a P2P overlay network are required to perform processing of data transferred across that network. Data can consist of either routing information (otherwise known as ‘maintenance’ traffic) or content storage/retrieval network traffic. Depending upon the routing algorithm implemented, the processing overhead impact will vary. For structured routing algorithms greater processing demands are assumed. Implicit within the implementation of a P2P overlay is the requirement for a constant data connection. When a node is not making demands of the network itself through a content request it could be servicing routing information requests from other nodes. Within a mobile domain, this software requirement alone impacts upon the relationship with the host hardware and the mobile network that it interacts with.

3.1.1 Significant Influencing Factors

With an assumption that enough processing capability within smartphone devices now exists [121, 32, 128], other restrictions that affect a node’s capability to perform on a P2P network have become higher priority. The most significant examples of these restrictions are listed below. Justification for concentrating on these influencing factors in particular is due to their effects on the introduction of P2P technologies into a ‘real world’ mobile domain.

- Poor battery life for node host device (Section 3.1.1).
- Data consumption (Section 3.1.1).
- Limited user interface functionality on node host device (Section 3.1.1).
- Variable data network connectivity and latency (Section 3.1.1).
- Compatibility with core mobile networking protocols (Section 3.1.1).

Influence: Poor Battery Life

The shortfall in adequate energy storage is a hugely important aspect of mobile device operation that is the focus of academic and manufacturer research and development programmes [170, 82, 81, 67]. Manufacturers are acutely aware that as the complexity and processing power increases within handsets, the demands placed on the battery become more significant. Consumers have come to expect devices that will operate for a time period measured in days rather than hours. With no significant improvements in battery life expected in the short term [96], the emphasis has been switched to optimisation of software operating on the devices as this correlates to device operating time. Supported by advice from mobile device manufacturers, developers can adopt various software operating strategies to minimise hardware energy consumption [13, 54]. Examples of these strategies include running efficient code to minimise CPU usage and minimising use of energy-intensive functions such as GPS, Bluetooth and the backlight. Most pertinent to the operation of a P2P node however was the recommendation to minimise data transmission as the transmitter is the largest consumer of energy within the handset. Transmitter usage in this instance relates directly to the data conveyed from and through the handset.

Influence: Data Consumption

The target of any node participating within a P2P overlay network is to minimise data consumption where possible. Minimising data consumption is an important factor to consider as it inversely relates to the length of time a node can participate with the P2P network. Data consumption on a handset relates to both the network activity generated through a data storage or retrieval request as well as overlay maintenance traffic. Hence it does not necessarily relate to the activity of the handset owner. Choice of an overlay implementation can affect both of these parameters although the constant nature of maintenance traffic means the latter will have a more dominant, cumulative effect overall. Minimising data traffic also has a commercial incentive for both the user and network operator. Although the cost of mobile data consumption is lowering [124], prices are not considered cheap and therefore users will not want to pay for excessive data costs caused by inefficient or data intensive applications. Network operators too have a commercial interest in data traffic remaining as low as possible: primarily for reasons of capacity and service quality influenced by increasing numbers of data consumers placing heavier demands on their networks [3].

Influence: Limited User Interface Functionality

Reduced capability places indirect requirements on software developers to write more autonomous applications. A change in thinking by designers and users alike from ‘impoverished to extraordinary interfaces’ as proposed by some academics [69] reflects the new approach to using mobile devices. With the increased complexity of software, users have also become more demanding in their expectations that it should be autonomous and self configuring. Users more frequently expect software to ‘just work’ or follow a simple set-up procedure. Therefore self (or minimal) initiation, configuration and maintenance should be the design goal of any mobile software application. The applicability of P2P systems in a mobile domain appear appropriate as existing P2P systems partially meet this requirement already through their normal behaviour.

Influence: Variable Data Network Connectivity and Latency

The performance of the handset should not be considered in isolation; the convenience and flexibility that a smartphone offers is in part due to the mobile telephone network it is connected to. However, the aims should be to minimise response time and maximise content availability. Allowing the host application quick access to the content data in the overlay network gives the user a more interactive and reliable experience, thus encouraging continued usage of the application. Although network speeds have increased from KBps to MBps, making for a more interactive user experience, network data speed is only part of the equation. Connectivity capability is also responsible for user adoption rates and should be factored into the requirements for any P2P network overlay. Having increased coverage from solely major urban conurbations in the early years of mobile networks, operators such as ‘Three’ (the UK’s largest 3G network operator) [63] quote population coverage in excess of 93% and increasing. Whilst network no-operation zones (or ‘black spots’) do still exist (e.g. sparsely populated regions of Scotland [105]), this technical inhibitor to P2P overlay adoption has largely been removed. Therefore whilst the likelihood of a particular handset not accessing the network over a wide area is small, there is still a likelihood of localised black spots due to terrain and building arrangements. A larger impact on data connectivity however will more likely occur from normal operation of the handset through handling telephone calls. Due to the operating characteristics of GSM technology, a data connection is dropped for the duration of a voice call when it is conducted from a handset. Therefore nodes must be capable of handling dropped network data connections through automatic reinstatement. Reinstatement of the data circuit can occur either a few seconds, minutes or hours later when the handset has either moved and network connectivity resumes or the voice call has ceased.

Influence: Compatibility with Mobile Networking Architectures

Any P2P mobile overlay is subject to network configuration practices and the restrictive constraints implemented by the mobile network operators. Although no public policy documents are made available to users by network operators on the subject of data network configuration, it is realistic to assume that in addition to basic technological constraints, security and simplicity are at the core of their design decisions. Technological constraints in this instance refer to Network Address Translation (NAT) [151], Firewalls and the resultant provision of private IP addresses for telephones on the operator network. These common networking technologies are configured so that that each handset functions on its own sub-network where there is only one device - itself. Connection of the handset is therefore only possible with nodes on the 'open Internet' via the network operator's proxy server. With no inter-device communication permitted within the operator network boundary, security is increased as it removes the risk of unauthorised access or virus transfer within the operator network. Billing functionality is also simplified because, using a client-server model approach, data is only sent to a handset that has been requested by that handset. This matches the common revenue model of a user only paying for what data they use. P2P systems in contrast can have data flowing into and out of a node independent of its own activity which is in conflict with this revenue generation model. The strength of this inhibiting factor has weakened in more recent times as 'unlimited' data consumption tariffs have become more prevalent within the market place. Further discussion on the subject of NAT, Firewalls and private IP address routing is presented in section 3.3.

3.1.2 Summary of Requirements for a Mobile P2P Overlay

Choosing a suitable P2P overlay upon which to base a mobile variant not only requires consideration of the general influencing factors already discussed in the previous section but closer evaluation of the specific properties highlighted in chapter 2. Defining a set of requirements goes beyond simple statements, it is also the production of a balanced view between opposing ends of a design spectrum: 'rigidity' and 'flexibility'. Constraining a design to rigid requirements ensures it meets a specific purpose or specification, for example overlay node capacity or response latency. Flexibility however ensures creativity can flourish in order to produce many viable solutions. In essence the requirements are to not solve the problem but to clearly summarise the needs that the design is to satisfy [74]. The numbers of requirements explicitly described will also vary depending upon the design goals of the process. Given the research aspect of this investigation, it was decided to promote generality and flexibility over rigidity within the requirements criteria. Selection of an overlay design should be dependent on its ability to match to the following four criteria:

- **Minimal processing overhead for host node.** Minimising load implications will have positive impact on responsiveness of host handset and battery life (due to reduced energy consumption).

- **Minimal network data traffic.** Positive cost and battery life implications by reducing network data traffic to a minimum.

- **Low latency time for content requests.** Responsiveness of application is improved, providing a better user experience.

- **Adaptable and secure design configuration.** An adaptable overlay design promotes adoption by software developers due to the wide range of application scenarios. Adaptability also ensures longevity within the industry and commercial market place, promoting increased reliability through constant refinement. Business market reports [91] also indicate that security is a prominent concern with users. Within this P2P system, security can refer to reliability and integrity as well as confidentiality of data. Longevity increases with confidence in the security of a system.

A guideline for implementation should be that behaviour should equal or exceed the current typical operating characteristics of a mobile client-server application scenario. This is the necessary and realistic guideline if the P2P framework is to achieve adoption from both developers and users alike.

3.2 Choosing a Mobile P2P Overlay

From section 3.1, the broad requirements for a suitable P2P overlay are applied to those technology categories broadly discussed in chapter 2. Through considered evaluation, unsuitable technologies are dismissed and further elaboration of suitable technologies is presented. Technologies have been considered on the basis of classification structure and the resulting sub-classification(s).

3.2.1 Elimination of Unsuitable Classification Candidates

Overlay Class: Unstructured vs. Structured

The choice of overlay structure style was the first and most fundamental design decision considered for the project. Both styles of overlay have proven operational heritage within the desktop computer environment. They are believed, in principle at least, capable of operating in a mobile environment. Justification for this premise included anecdotal evidence of smartphone handset processing capability and readily accessible internet connectivity (see section 1.1). Prediction of P2P networks within a mobile scenario was also envisaged by Lua et al. [62] as a direction for further research. Unstructured P2P networks have the least complex routing algorithms of the two styles and thereby the likely lowest computational processing overhead. This would be considered an advantage when operating with the limited processor resources of a mobile telephone handset. However, certain fundamental characteristics of unstructured P2P networks could be attested as less suitable for implementation on this project than a structured P2P network. More specifically, unstructured P2P networks suffer in terms of data search and storage efficiency. A clear statement by Cohen et al. supports this assertion [114]: 'Search is blind in that it is independent of the query and is thus not any more effective than probing randomly chosen peers. One technique to improve the effectiveness of blind search is to pro-actively replicate data.' However this has to be balanced as excessive data replication causes wastage of network resources [111]. By comparison, the advantages offered by using structured P2P overlays present rewards with much greater user impact. Most significantly, using a structured overlay ensures that node look up process or data discovery is deterministic [165]. Knowing that data can be explicitly found or declared 'not present' is important when limited resources such as bandwidth and battery power have contributed to the search request. Structured overlays, through their 'locally stored' network routing knowledge, will typically minimise network traffic and offer improved response times (lower overlay latency) in comparison to unstructured overlays when retrieving the desired data. Additionally, minimised network traffic will contribute to longer battery life and potentially reduced operating costs for the user. Having opted to select an overlay routing algorithm from the structured genre, further consideration was required to select an appropriate sub-category of type and implementation.

Structured Overlays: Single vs. Multi-hop

From chapter 2, when considering types of structured overlay the options available were related to the number of hops and topology design. Whilst a single hop overlay would offer the lowest latency time for retrieving content from the network, the case for lowest network traffic was less clear. Single hop networks perform at their best when the churn rate of the nodes within the network is low so that the routing table that maintains data for the entire network remains 'fresh' and accurate. This scenario is unlikely to happen within a mobile environment where nodes are subject to many varying conditions that will affect their connectivity status. For example, network availability, user data tariff or battery life might affect a users decision as to how long their handset participates within the P2P network. Evaluation of variable hop networks has indicated that, whilst a possibility for future consideration, the existing technology is not thought to be mature enough to warrant an investment of effort for uncertain returns (especially in comparison to multi-hop overlay networks). Therefore a multi-hop routing network overlay became the clear category from which to either develop or select an appropriate algorithm. Selection of a suitable overlay solely on the basis of topology design rather then routing implementation was considered less appropriate given that the routing algorithm would more closely dictate the levels of network data traffic.

Multi-hop Structured Overlays: Choosing a candidate

With reference to section 2.4.2, several multi-hop algorithms are available for consideration. All four of the implementations discussed, Chord, Pastry, Tapestry and Kademlia, have significant and proven operational heritage. For example Kademlia is the underlying overlay technology implemented within Mainline DHT, the largest DHT on the open internet [51, 148]. Any one of these could be capable of being implemented within a mobile domain with differing levels of suitability and success. All four technologies presented multiple sources of academic and commercial investigation to warrant explicit consideration in the next section (3.2.2). Each of the technologies will be compared and contrasted against the primary defining aspects of structured P2P overlay design.

3.2.2 Elaboration on Suitable Classification Candidates

Chord, Pastry, Tapestry, CAN and Kademlia all could be considered suitable for application to a mobile network domain. All present a structured, deterministic, routing geometry to the network overlay. A deterministic approach ensures that object location can occur if it is present as each node is aware of its position within the network. Symmetric behaviour occurs within the overlay design because each node is of equal importance to other nodes within the network. When considering data being stored in either of these overlay networks, no difference is made between actual data and 'pointers' to data (such as URLs). The latter may be preferred within a mobile domain given the data storage constraints of mobile devices.

When considering all of these algorithms, evaluation against a set of specific criteria aids drawing out a comparison for deciding which performs best in the mobile scenario. Buford et al. [65] performed such a process for the comparison of a number of algorithms. This process included evaluating quantitatively parameters such as convergence, stability, fault tolerance and load balance. Another comparison was conducted by Dhara et al. [165] comprising a review of the key design parameters of twelve algorithms including the four under focus in this chapter. The comparison presented within this chapter has utilised key parameters discussed in both of these reviews. Forming a mixed qualitative and quantitative comparative review, the algorithms were evaluated against the following key properties:

- Routing table design. (Section 3.2.2)
- Routing distance metrics. (Section 3.2.2)
- Routing dimensions. (Section 3.2.2)
- Routing table maintenance traffic. (Section 3.2.2)
- Bootstrapping. (Section 3.2.2)

Each of these key properties for a structured overlay can have varying degrees of suitability for operating within a mobile domain. Discussion surrounding this subject is presented in sections 3.2.2 through to 3.2.2.

Routing Table Design

The incorporation of a DHT within routing table design was introduced in chapter 2.4.1 with both Chord and Kademlia. As previously highlighted, the use of DHTs ensures there is a determinable process of finding data stored on nodes and basic load balancing of the content across the overlay network. Both overlay technologies build their routing tables adaptively as either their knowledge of the network or the number of participants within it expands. This process involves assimilating information on surrounding nodes either by explicit querying of network neighbour nodes or monitoring of through network traffic for information (particularly when a node joins / leaves the network). In both cases, routing information consists of unique node identifier details with respect to both the overlay and physical network. However, the routing table design and operating algorithms differ between the overlay technologies. The following subsections introduce the salient aspects of routing table design for Chord, Pastry, Tapestry, CAN and Kademlia.

Chord: Chord utilises a routing algorithm similar in functionality to a traditional search Skip List. Skip Lists utilise a balanced tree hierarchy of linked lists in order to efficiently link increasingly dispersed data from the searching node perspective [159, 136, 94]. This general overview of the node structure is supplemented by more detailed information obtained from nodes 'closer' to the target node in the logical ring. A search is taken down through progressively more detailed lists of identifiers by querying continuously closer nodes until the desired parameter is found. Within Chord, the lists are held within the routing tables (also known as Finger Tables) and contain lists of sequential node identifier information [18]. Each node's routing table only possesses information on certain subsets of all the nodes within the network. The Chord algorithm ensures these subsets are organised with a higher number of nearby node identifiers being known. Chord can operate with some degree of node churn because it must ensure that the node's successor pointer (i.e. the link to the next logical node in the list) is kept up to date. These points are illustrated in general terms within Figure 3.2 where the searching node knows of identifiers (of successor nodes) in surrounding positions 2^n steps away from itself in the logical ring. The longer a node remains active within a network, the more knowledgable the surrounding nodes become of it. To increase the robustness of the network further, each Chord node maintains a list of successors so that multiple nodes can be contacted should the first successor not be available due to node failure or departure. The performance of Chord is also dependent on the size of the network; the node lookup latency grows as the number of participating nodes increases. Chord's look up latency can be significant for another reason: physical network latency. The node identifiers are randomly distributed across the identifier space. Whilst the search source and target can be 'logically close' within the identifier addressing space, they can be far apart in the underlying physical network.

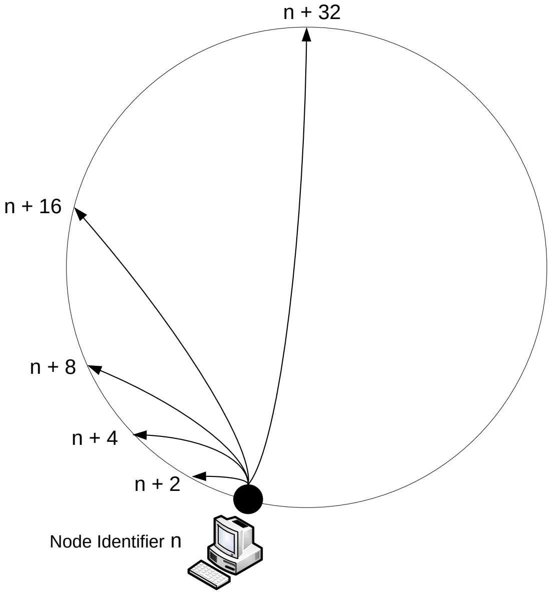


Figure 3.2: Chord Overlay Routing Design

Pastry: Pastry utilises a routing algorithm which is based upon the numerical proximity between specific node identifiers. Routing tables constructed using the Pastry algorithm achieve a similar result to those seen in the Chord overlay. The node identifier is typically based upon a secure hash of a public key assigned to the node, such as the network IP address. This again is common with other overlays in order to utilise the property of uniform dispersion across the node identifier address space. The routing state awareness of a Pastry overlay node is divided amongst three distinct elements: Routing Table, Leaf Set and Neighbourhood Set. The Routing Table stores all known links to nodes in the overlay identifier space whilst the Leaf and Neighbourhood Sets maintain lists of nodes that are closest in terms of node ID and network locality respectively. The Leaf Set (similar to Chord's successor list) helps to improve the look up efficiency whilst the Neighbourhood set supports routing decision by working with the Routing Table to maintain a more accurate, fresh view of the network locality. The routing table behaviour is similar to Chord in that routing table data is more detailed for nodes closest numerically to the table host node within the network.

Tapestry: The Tapestry algorithm has different origins to the other technologies considered. Based upon the Plaxton routing algorithm [98, 133], it can be considered as a mesh rather than a ring. The node location mechanism operates by sending a message between source and destination node through a root node. However, unlike Plaxton, Tapestry has multiple root nodes to serve the purpose as the target of node searches. The Tapestry algorithm is designed such that every node within the network has a 'neighbour map' style routing table. Tapestry has an explicit notion of its network locality. Tapestry also has several aspects of its functional behaviour that are adaptive in order to increase its operational efficiency. These include: using a routing table that has multiple levels of granularity; keeping the routing table fresh by recording data passed through it by other node searches; monitoring the frequency of requests made to a particular data object; and caching additional copies of it across the network if it appears popular. Tapestry can also maintain its routing table by actively removing nodes it believes have failed by monitoring network link timeouts from node routing traffic data.

CAN: The CAN algorithm is contrasting to other technologies at a more fundamental level: a different design approach associated with node organisation within the overlay. Rather than distribution of nodes across a logical ring or mesh it is across a multi-dimensional cartesian coordinate space. Nodes are are addressed as ‘zones’ (as indicated by letters within Figure 3.3) dispersed across the complete cartesian coordinate space, and the structure is independent of physical connectivity and location. Node zones logically vary in size due to the node join algorithm which is dependent upon where another node joins the overlay within the coordinate space. Navigation between nodes uses a combination of IP address and routing zone coordinates which are stored within the routing table. A routing table for a particular node is constructed from only the coordinates of its adjoining nodes in the address space. Therefore with a d dimensional coordinate space the routing table need only maintain data about 2d neighbours. The constrained routing table intuitively supports direct (or straight line) routing across the co-ordinate space as Figure 3.3 illustrates. The routing path however is not constrained to one particular passage and a CAN overlay will offer varying routes dependent upon the nodes (zones) present at that time. For example, within Figure 3.3 routing between zone G and M is shown via zones J, H but could it also be via zone F.

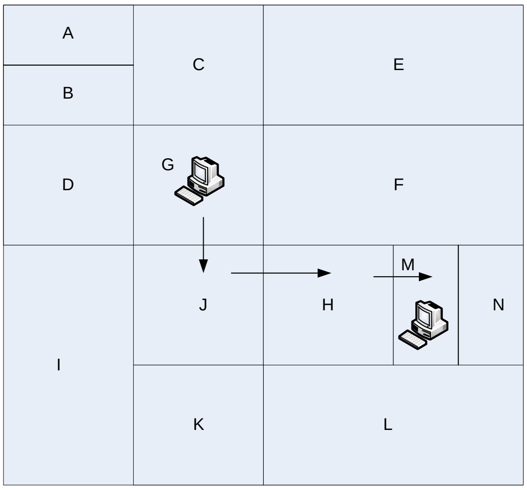


Figure 3.3: CAN Overlay Routing Design

Kademlia: Kademlia employs a ‘tree style’ routing algorithm where routing information about nodes is split into branches dependent upon the prefix of the node identifier. Within the Kademlia overlay, each branch is known as a routing ‘Bucket’ as illustrated within Figure 3.4. One bucket is associated with each bit of an identifier address, so the identifier size (typically 128 bits) directly affects the routing table size. Within this illustration a 4 bit NodeID will give a maximum of 16 buckets which can be occupied with NodeId / IP address data in the routing table. The main advantage of the tree structure approach is speed; for each bit that is analysed, the search process only has half of the remaining nodes known in that routing table to consider. Searching for a node consists of splitting its identifier address into progressively smaller sections, each of which links to a progressively smaller sector of the address space. If the required node is not known in a particular bucket of a routing table, then the search query is forwarded to another node that is closer to perform a search. If the desired node is present on the network then eventually, through progressively finer search steps, the search will return the correct routing information. This design approach works because the Kademlia algorithm ensures that a node always maintains more routing table information about nodes closer to itself in the logical address space (i.e. similar identifier address) than those further away. The search query process helps to permeate details of a node’s existence within a network as its node identifier is broadcast along with the query. This information is then used to populate the routing table of any queried nodes regardless of whether it has the desired search results. This is not the only process used to keep routing table data recorded on each node accurate; a ‘refresh’ procedure is performed periodically by contacting nodes individually to determine if they are still present on the network.

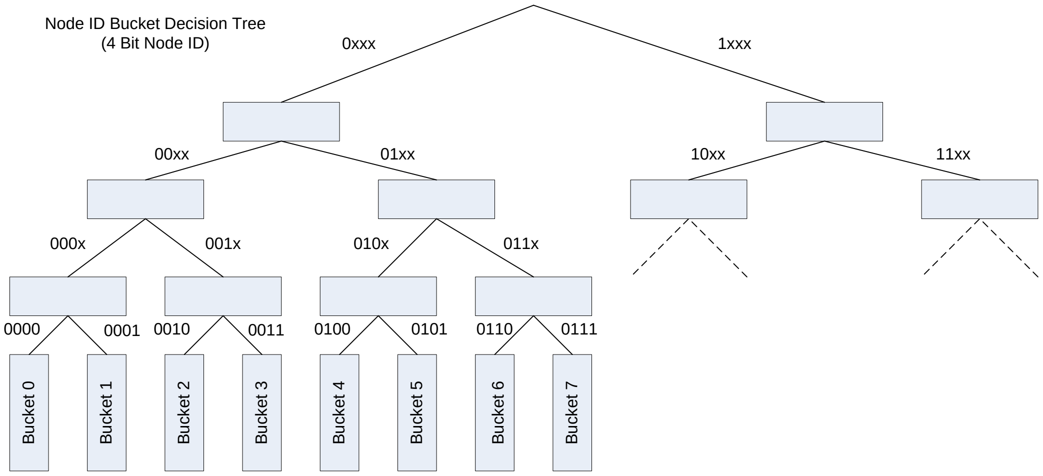


Figure 3.4: Kademlia Routing Table Design

Routing Distance Metrics

All overlay algorithms operate under the conception of ‘proximity routing’: routing utilising a notion that a search message will get ‘closer’ to its target on each successive network node hop. These overlays have the idea of ‘distance’ or ‘closeness’ integrated into their routing logic. Distance in this instance does not refer to physical distance of nodes but rather a logical abstraction associated with the routing algorithm. This metric is used to conceptually determine how ‘far apart’ nodes are within the address ID space. By reducing this metric on successive forwarding steps within a network, an algorithm can deterministically ensure it is getting ‘closer’ to reaching the search objective. Distance metrics are used to determine routing parameters (e.g. the number of node hops required to access the information) and routing directions (e.g. for data placement and search requests). Each node within the network will determine its ‘distance’ from another node using a pre-defined function that is implemented across all nodes within an overlay. Chord calculates its distance metric as the numeric difference between two node identifiers. Pastry incorporates the concept of closeness of identifiers through the explicit use of defined prefix and suffix structures, focusing primarily upon the prefix. The numerical proximity (or ‘distance’) metric is typically generated using data readily available from the network infrastructure such as number of IP address hops, round trip message delay or geographical location of nodes. Tapestry differs in this respect by having routing dependent upon the suffix of a nodeID. Tapestry also places references to the desired data on intermediary hops between the server and root nodes. This is in contrast to Pastry and other algorithms where direct routing using the nodeID to the network vicinity holding the data is the desired operation. Due to the uniformity of the search space, CAN has an easily predictable average routing path length. This path length relates to the number of dimensions of the cartesian coordinate space and number of zones that it is currently split into. This can be summarised by the following equation with *d* dimensions and *n* nodes [115]:

$$\left(\frac{d}{4}\right) * \left(n^{\frac{1}{d}}\right) \tag{3.1}$$

Within Kademlia, it is a bitwise exclusive OR (XOR) of the two node identifiers concerned. Although different for each overlay, neither algorithm offers any significant operational advantage over the other.

Routing Dimensions

Following an introduction in Chapter 2, another abstract concept that is linked with routing distances, is the idea of an address space. The address space of most P2P overlays is typically considered as a logical circular ring (i.e. nodes arranged in a ring in order of their identifier). These rings can have dimensions as well, typically one but there can be more when super peer node rings are considered. Chord, Pastry and Kademlia address spaces are considered as a 'flat, logical ring'. 'Flat' in this instance refers to one dimension and is as illustrated within Figure 3.5.

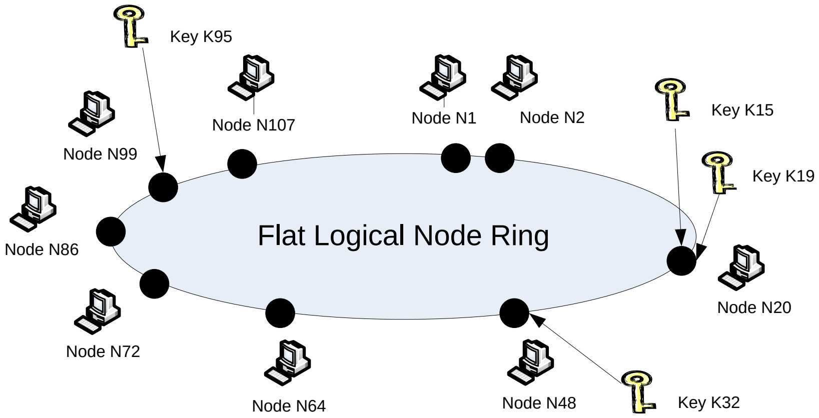


Figure 3.5: P2P Address Space: 1 Dimensional (Flat), Logical Ring [7] (Adapted)

Within this figure, node identifiers are depicted as 'N' numbers whilst data identifiers (or Keys) are shown as 'K' numbers. As this diagram also illustrates, both these identifier number sequences share the same address space. In accordance with the Kademlia overlay algorithm, when there is no matching node identifier for a key, the nearest logical node is selected as a substitute. Tapestry operates under a different routing dimension design: a mesh design where each node has a 'view' of and accessibility to the network locality surrounding it (see Figure 3.6). CAN operates, as has been introduced, within a 'd' dimension cartesian coordinate space. This space can be thought conceptually as flat (2d) or as a d - torus (ring). All routing dimensional structures have advantages and disadvantages primarily concerning complexity and scalability. Upon initial consideration the mesh topology offers maximum flexibility and routing security as data will always find a path from source to destination given a complete mesh. However, given the number of interconnection permutations, serious concerns have been raised about maintaining such a complex network when the node volume scales up. Additionally, problems have been mooted concerning routing response times given the number of intermediary nodes in a data retrieval request. Logical rings however can maintain scalability by keeping routing complexity within the same order of magnitude by focusing routing data on the section of the ring that a node joins.

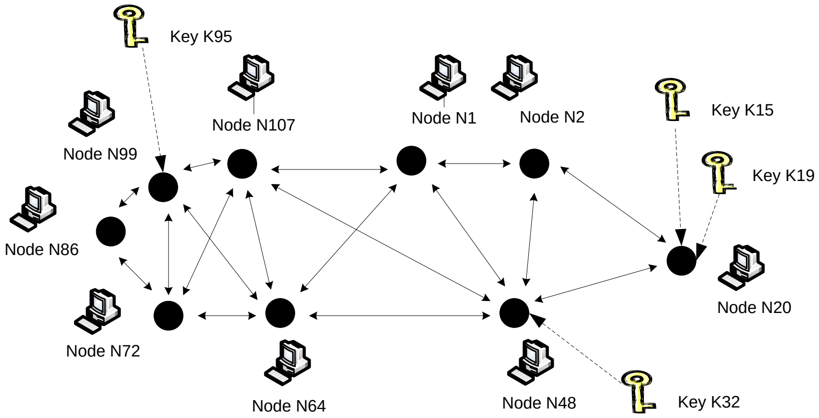


Figure 3.6: P2P Address Space: Mesh Topology [100] (Adapted)

Maintenance Traffic

Chapter 2 introduced the concept of maintenance traffic: data that conveys routing information for recording within each node's routing (or finger) tables. Maintenance traffic in essence constructs the overlay design implemented within the algorithm and is intrinsically involved with the stability of the overlay. Whilst this data traffic across the network is fundamental for P2P systems, the overhead imposed by this can be significant depending upon the algorithm implemented. Maintenance traffic can vary in terms of volume of data moved and timing frequency. The traffic can be determined by the size of routing table and required data retrieval latency as specified by the P2P overlay in use. The choice of maintenance update procedure also affects data rates and volume. These parameters are significantly influenced by the rate of churn of nodes within the network. Churn rate can also influence the choice of overlay as it affects responsiveness and accuracy of the results depending on the size of routing tables. For an overlay design with small routing table capacity and experiencing high node churn, the maintenance traffic profile on the network will be low quantity and frequent. In contrast, for an overlay with a larger routing table capacity and lower rate of churn, the maintenance traffic profile will be larger and more infrequent. Two principal maintenance update algorithm categories are implemented within an overlay structure, commonly known as 'active' and 'opportunistic' [165]. The 'opportunistic' process operates by inspecting routing data passing through a node either as standard network data requests or responses. Whilst the overhead of this approach is minimal, a drawback is that this approach works best on a network with regular data activity as it depends upon a flow of data requests and responses to keep the routing tables up to date. In contrast, the 'active' process operates well regardless of network action as a specific aspect of the overlay algorithm is dedicated to propagating routing table information across the network. The overlay designs discussed so far all employ approaches that align themselves with both processes. Chord employs an algorithm to run periodically to assess the status of the surrounding network whilst Pastry nodes periodically query their neighbours, using 'keep alive' signals so it can update its tables. Tapestry nodes broadcast to their surrounding nodes notification of their pending departure when leaving the overlay network. Whilst only possible when a node departs a network gracefully, this approach will help maintain fresh routing table data. CAN overlay networks and routing tables are maintained primarily through two mechanisms: inspection of CAN messages being transmitted through nodes and graceful departure of nodes from the overlay. Notification by a node to its neighbours when departing the overlay and the subsequent transfer of zone responsibility to another node help greatly in maintaining an accurately recorded overlay infrastructure. Nodes within a Kademlia overlay network however take a different approach; a node will periodically refresh its routing tables by using a sequence of timeout controlled 'ping style' messages. These messages will test the presence of nodes within the overlay; any unresponsive nodes will be removed from the routing table. Kademlia can also conduct node searches in parallel, for example by querying all of the nodes within a tree sub-branch. The degree of parallelism for a search (i.e. how many simultaneous node searches are underway) is referred to as the α (alpha) value [135]. The linear and parallel 'node look-up' approaches are opposing search philosophies. In a 'real world' implementation of a Kademlia system they are traded against available node processing power and accessible network bandwidth.

Bootstrapping Procedure

Whilst Bootstrapping has a generic computing definition, it is also terminology relating to the start-up behaviour of a node within a P2P overlay network. Defined as the operation that is executed when a node first joins a P2P network [165], successful bootstrapping algorithms are vital if a node is to become active within the overlay. Several approaches to bootstrapping exist, but all with the strategy of supporting peers in connecting into the network quickly [165]. One approach is for nodes to connect to a pre-defined server that maintains relatively fresh node routing data to permit initial population of the new nodes' routing table. Another variation on this approach is for a new node to connect to one of a pre-arranged list of existing network nodes (where at least one of these will always be visible upon the network). Bootstrapping is an important parameter within P2P overlays as it not only fundamentally affects the ability of a node to join a network, but can affect a node's performance during the formative participation period. This latter point is more important if the node churn is high as the node will have a shorter time to become accurately connected into the overlay for productive activity. With reference to the overlay technologies discussed previously in this section, Chord and Kademlia overlays require being aware of an existing node on the network for their bootstrapping regime. Pastry and Tapestry however only have to be aware of 'nearby' (i.e. logically close in terms of a proximity metric) nodes in their overlay to establish a link. Knowledge of the existing nodes is obtained through a proximity metric and contact established with the intention of contacting other local nodes. With inter-node communication established, all nodes that should be aware of the newly joined node will eventually have their routing tables updated. CAN operates by a node selecting and joining another known node on the overlay and then taking on responsibility for half of the existing nodes' zone within the search space. This action prompts the new node to become aware of its neighbours within the overlay.

3.2.3 The Chosen P2P Technology

Due to the funding mechanism for the research, as introduced within section 1.1, choosing an overlay for a practical implementation required balancing priorities between academic and commercial criteria. The academic criteria are a qualitative assessment of general performance within categories discussed under section 3.2. Judgements were also made on the overlay's future applicability to a mobile scenario. Section 4.1 outlines these in more detail. The applicable commercial criteria are an assessment of compatibility with the aims outlined at the start of the chapter. The P2P overlay Kademlia was selected for pragmatic reasons against both these measures.

From a technology perspective, discussions under section 3.2 show Chord, Pastry, Tapestry, CAN and Kademlia have many theoretical and operational similarities. For example, with reference to bootstrapping and maintenance traffic, given the similar implementation design philosophies Kademlia's performance is not significantly weaker or stronger than the other designs discussed. Kademlia exhibits a positive characteristic of promoting stability within a peer network by ensuring that 'every message exchanged conveys or reinforces useful contact information' [71]. Others designs, such as Chord do not and this could have an impact on reliability within a network of significant node churn. The authors of the original Kademlia work [71] indicate that the routing table design of Chord in particular leads to 'rigid routing tables'. However, some results from the comparative simulation studies by Buford et al. [65] show that the Chord algorithm performs marginally better on a cost-performance comparison than Kademlia as Kademlia's routing table on the nodes may be larger than Chord's. This implies an improved lookup performance, but requires a slightly higher maintenance regime.

One further favourable Kademlia operating characteristic is that when a node leaves the overlay there is no requirement for it to notify any peers. This aspect of the overlay functionality supports the envisaged mobile network behaviour of significant node churn levels. Peer churn is handled efficiently within Kademlia by maintaining details of several node neighbours for each routing table entry. They are ordered by length of time known with details of newer peers replacing existing neighbours when necessary, thereby 'mitigating effects of high infant node mortality' [65] which otherwise could be very disruptive in a mobile environment. As Maymounkov et al. state [71], 'Kademlia is the first P2P system to exploit the fact that node failures are inversely related to uptime'.

Another justification for selecting Kademlia was the minimal processing overhead envisaged with its routing algorithm and the simplistic nature of its routing table data organisation. The authors of the original Kademlia work [71] indicate that another strength of Kademlia is a single routing algorithm 'from start to finish'. Complexity is therefore reduced in comparison to technologies such as Pastry where two are used depending upon the stage within the routing process. Based upon division into 'buckets', routing data is grouped by binary similarity of the ID working on each individual digit. The net effect is to halve the search space for each binary digit in the search ID successfully matched to the target ID. The relatively simple routing algorithm, based upon an 'XOR' logic comparison of a node identifier and key term, was considered within the processing capability of existing commercially available handset devices. This assessment was supported by exploratory work conducted at the Budapest University of Technology and Economics (Department of Automation and Applied Informatics) in association with Nokia and Siemens Networks [53]. This collaborative research partnership had produced (and made freely available through an open source software licence) a 'proof of concept' implementation of a mobile variant of Kademlia. The mobile Kademlia software [8, 28], whilst not fully functional, did offer adequate qualitative evidence that the processor capability of the handset could handle the load presented by the Kademlia routing algorithm.

Discussions presented over the previous sections have shown that, whilst several overlay designs existed, all had performed the required task to a similar level. The pragmatic outcome of these considerations therefore was that no particular technology had a significant advantage over the other. Given this outcome, assessment against commercial criteria then became influential in the selection process. Kademlia did possess an advantage that it was not directly linked to a commercial research organisation and therefore the design is likely to remain 'open and accessible'. The seminal academic paper [71] was part funded by The National Science Foundation, an independent United States Government agency with educational aims. This is in contrast to Pastry which was funded initially by Microsoft Research. Also the availability of a ‘proof of concept’ application [8, 28] for the Nokia smartphone platform did attract the attention of the project’s economic sponsors due to its obvious synergy with the project’s requirements. Having legally accessible and working source code provided foundations for meeting the requirements outlined in section 4.1 whilst also complying with the project’s commercial time constraints. Therefore the decision to use this prototype application implicitly meant selecting Kademlia as the P2P overlay. In essence, a project decision was made to gain commercial expediency without compromising on the technical solution. Further details of this software application, including its features, limitations and the implications on the work produced for this project, are given within section 4.3.

3.3 Working Within Network Constraints: NAT Traversal

This section discusses the issues and constraints as well as hardware and software requirements associated with the provision of a functional P2P network, accessed through a telephone network operator internet gateway.

3.3.1 NAT Concepts

Network Address Translators (NAT) provide both the necessary functionality and a substantial barrier to interactive communication in a wide area network of nodes. One of the original design premisses of the Internet was uninterrupted, direct communication between all internet enabled devices. This was idealised as one ’public’ IP address per internet enabled device. However, the idealised configuration of early implementations has required modification to cater for ever increasing numbers of connected devices. Network address translation, in generic terms, is the process of assigning multiple computers on a private network to a single, publicly visible IP address. In essence, NATs aid the sharing of a limited public IP address resource. Some estimates indicate over 70% of hosts on the internet are located behind a NAT enabled device [65]. This translation process became necessary because the volume of internet enabled devices was predicted to exceed the availability of publicly accessible IP v4 addresses by 2009 according to IANA / ARIN [17, 147]. The volume of private IP addresses however is not an issue as they can be re-used simultaneously across many private networks invisible from each other by the masqueraded public IP address, implemented by the NAT. Unless network communication is performed solely within a single network domain, NATs will impact on the communications channel by breaking the functionality of the protocol (in this case, but not exclusively, P2P protocols). The process of overcoming the obstacles introduced as a consequence of NAT is referred to as NAT Traversal. NATs were one of the fundamental reasons for the slow adoption of VOIP technologies. This is because a significant proportion of VOIP users (85%) are connected to the internet through private networks [45] which access the internet via a NAT device. The problem has been alleviated by the introduction of IP v6 addresses, which give a larger pool of public IP addresses, but the presence of NAT will remain for the foreseeable future to support existing infrastructure. Re-usability of IP addresses is no longer the only reason for using NAT; security is also improved as the allocated IP address of a machine is hidden from the public network (and other private branches).

3.3.2 Defining NAT Traversal

As stated, NAT is the process of assigning multiple computers on a private network to a single, publicly visible IP address. This process obscures the internal structure of the private network as to all outside systems the traffic appears to originate from the network gateway. Figure 3.7 illustrates simplistically how similar IP addresses of devices on private home and office networks are all masqueraded by a single public IP address typically assigned by the Internet Service Provider (ISP).

NAT functionality is typically incorporated into routers, gateways or firewalls located at the ‘entrance points’ of private networks. NAT operates by replacing certain parameters of data packets on the private network, principally the private IP address of the originating device with the public address of the router/gateway/firewall. This device then takes responsibility for ensuring that any data packets received at its public IP address are routed back to the correct origin on the private network by replacing IP addresses in reverse. This process is performed by maintaining the current state of translation tables within the NAT enabled device. In essence, the communicating parties are not aware that they are using router masqueraded IP addresses. The NAT process does not solely work upon IP addresses; for more reliable results it has to be used in conjunction with port number mapping (via the TCP or UDP port information). Typically basic NAT operation will focus solely upon the IP address of machines.

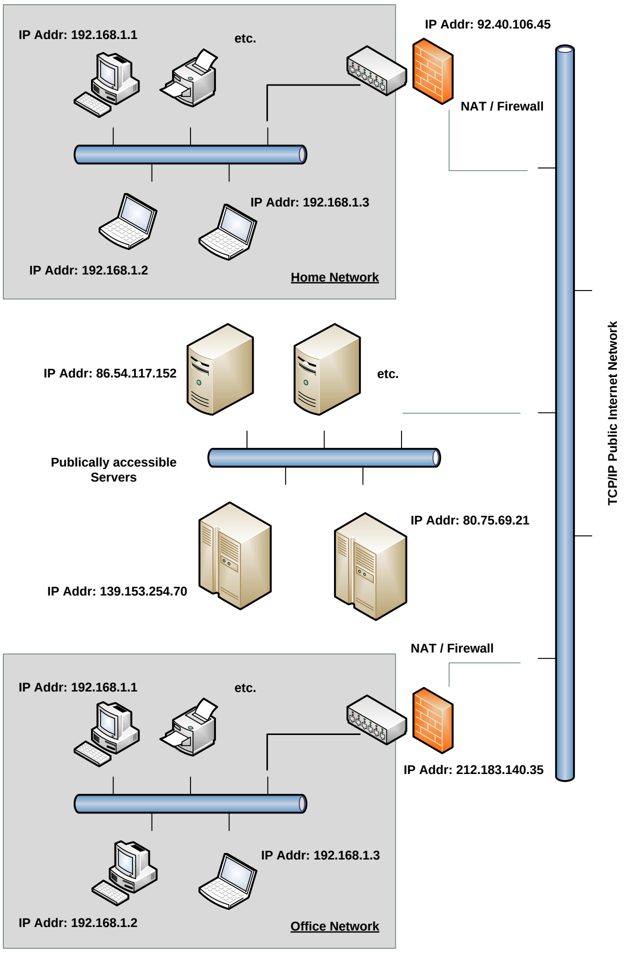


Figure 3.7: Example of IP Address Masquerading on Private Networks Using NAT Devices

However, Port Address Translation (PAT) is also possible and allows different applications upon a single machine to operate across the translator barrier. Another use for PAT is to permit the constant access of devices behind a NAT from devices in the public IP address space in a configuration known as ‘Static NAT’.

NAT does not pose any problems for a typical client-server relationship between a user device on a private network and a public server. This is because data is flowing in a single communications channel between the client and server; the server responds to a request from the client which is easily tracked by a NAT router. However the communication channels present within a P2P network are much more complex; network nodes act as both a client and server to each other. The traditional NAT approach breaks down when data is attempted to be sent to a network node that did not request it. The intermediary router providing the translation process cannot successfully forward the data as it has no knowledge of an outgoing request matching the IP address presented. The requirement for publicly reachable IP addresses then becomes clearer for a P2P network as these are the addresses stored within each node’s routing table.

3.3.3 Different Approaches to NAT Traversal

It is acknowledged that NAT traversal is problematic for P2P networks and yet is possible to overcome using certain technologies [163, 92, 102]. NAT technologies have evolved as the complexity of the internet has matured and can be categorised into several design groupings typically based upon their behaviour. Several methodologies have been devised to support P2P style network activity by circumventing the problems presented by NAT. These methodologies support all forms of interactive communication such as VOIP, IM and file sharing. The NAT process is however not consistent across all devices across the Internet and therefore traversal methodologies have had to be flexible to cope with this. The most significant approaches to NAT Traversal are summarised in sections 3.3.3, 3.3.3 and 3.3.3.

STUN: Simple Traversal of UDP though NAT

STUN [66, 160, 166], Simple Traversal of UDP though NAT (updated to Session Traversal Utilities for NAT), is an amalgamation of protocols and processes designed to determine the presence of a NAT. If a NAT is present then STUN will perform a public IP address and port number translation of nodes behind the NAT. The STUN protocol is designed to be lightweight to have minimal impact upon its host network but does require the presence of a publicly addressable STUN server. Based around the connectionless UDP protocol, a series of messages is sent between the client (on a private network) and a STUN server. The STUN server notes the IP address / port information received from the client and returns it via another message back to the originator. The information held within these messages then allows the client to determine the translation process its messages undergo in accessing the public network. From this process, the NAT-associated public IP address / port number can be determined as well as other relevant parameters such as a 'lifetime' figure for the port number. The node on the private network can then advertise its NATs public IP address and port number (associated with the node) rather than its private masqueraded details to other nodes to ensure messages are conveyed successfully. STUN is not always reliable at recovering IP address/port information for a NAT due to inconsistencies of NAT behaviour and the connectionless state of the UDP protocol.

During the early adoption stages of STUN technologies several variations in the design emerged based upon differences in the operating behaviour. For example terminology such as 'Full Cone' NAT, 'Restricted Cone' NAT and 'Symmetric' NAT were introduced to describe the data flow across the NAT boundary. Full Cone NAT behaviour matched the operation described in section 3.3.2 where data packet flow followed 'one to one' matching between any internal and external network device across a NAT device boundary. The NAT device would ensure the correct data would be routed between matched devices. Restricted Cone NAT referred to limitations placed on either address or port mapping, whilst Symmetric NAT described a packet flow that was restricted solely between specified nodes. Symmetric NATs enforce behaviour where a reply from a node can only be returned to the message sender. Furthermore, for each destination selected from a node behind a Symmetric NAT, regardless of source application, the resultant output port from the NAT changes. Both of these properties make this category of NAT one of the more challenging to traverse. Figure 3.8 illustrates simplistically two opposites of NAT design in terms of a standard client / server connection configuration: Full Cone where there are no restrictions on data flow and Symmetric NAT where it is restricted.

Unfortunately, confusion and inconsistent implementation of these behaviours has meant that these descriptions are no longer formally used within the original STUN standard specification. The operational paradigms they introduced however formed the historical foundations of more up to date and reliable NAT implementation schemes. Furthermore, this process is made more challenging when two NATs are encountered in a communication path such as for private network to private network communication. In this scenario the use of another intermediary proxy application server, referred to as a Destination NAT (DNAT), can assist in the address determination process.

TURN: Traversal Using Relays around NAT

TURN [70, 162, 166] is considered the most bandwidth intensive solution to ensuring a message is sent between communicating nodes. TURN employs a relay (or proxy) node with a public IP address that is capable of communication with nodes located on both public and private networks. The purpose of this node is to forward data between nodes located on private networks. Each of the nodes on a private network is unaware of the other's private IP address, only that of the (public) intermediary relay node. It is the relay node that maintains a look up list of the public IP addresses of the routers/firewalls/gateways it is in communication with. TURN has the advantage of providing communication where other protocols (such as STUN) fail, assuming that nodes located on a private network can send data to a public IP address. This apparent reliability comes at the expense of resources, principally network bandwidth and provision of a public addressable server. Bandwidth should be considered in two respects: double the network consumption resulting from sending the data twice (i.e to and from the relay) and the capability of the TURN server to handle all of the communications traffic.

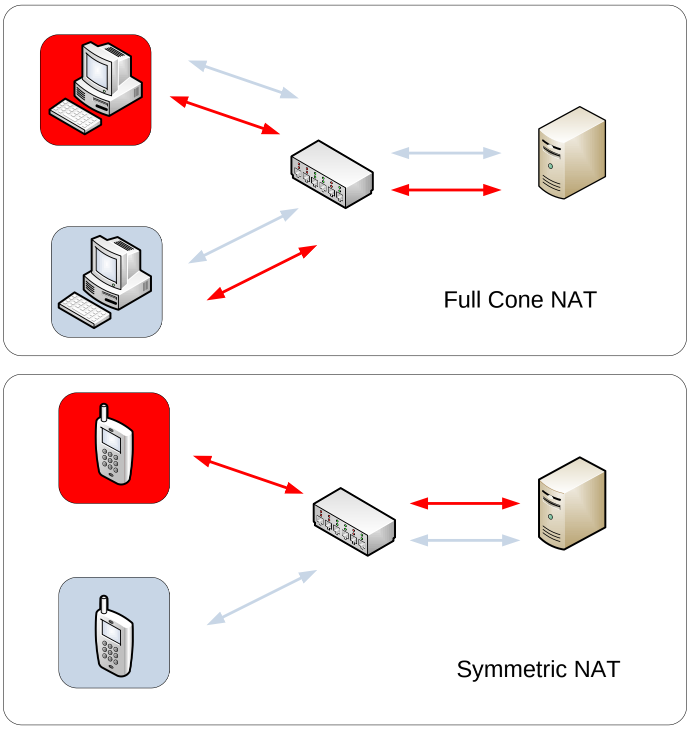


Figure 3.8: Illustration of NAT Designs: Full Cone and Symmetric.

ICE: Interactive Connectivity Establishment

ICE [101, 102, 145] is a P2P NAT Traversal framework created and currently under review by the IETF MMUSIC Working Group [27]. ICE seeks to coordinate the approaches taken by the STUN and TURN technologies in order to successfully determine IP address data across many disparate NATs. Consequently ICE implements an improved solution that will work with most firewall/router devices. Dissecting the communication via a NAT is first attempted with STUN as it has the minimum bandwidth overhead and has a greater probability of working with private, consumer networks. If this approach is unsuccessful or an Enterprise NAT is detected, the use of TURN is automatically employed to determine the address details of a particular node. The combination of both technologies in ICE acknowledges that there is some inconsistency about the implementation of NATs. However, the usage of ICE ensures the most optimal technology is applied to a situation.

3.3.4 NAT Traversal Solution Criteria

When considering the challenges faced within a mobile network domain, several similar criteria for selecting a suitable methodology arise as concerned with the choice of P2P network algorithm. In essence, implementing a solution that acknowledges the limitations of the technology participating within the system. Criteria for the mobile P2P framework included:

- *Simple, flexible and secure design configuration.* Simple, flexible solutions are most likely to be adopted and remain in a commercial scenario. When a design is kept as simple as the problem permits, issues such as security and maintainability are easier to confront.
- *Low data processing / forwarding latency time.* Responsiveness of a P2P network is improved, providing a better user experience, by keeping both processing requirements and data traffic overhead to a minimum.
- *Minimal network data traffic.* Positive cost and handset battery life implications are taken into account by keeping the data traffic overhead associated with this necessary technology to a minimum.

The solution to be implemented within the Bushfire mobile P2P framework is proprietary, a variation of the TURN design discussed. Built primarily through necessity the solution, code named ‘SysProxy’, sought to meet all of the above requirements. As with other systems, the SysProxy solution makes use of a relay or proxy server located on a publicly accessible IP address machine. SysProxy does differ from the technologies discussed by operating with TCP protocol data rather than UDP, primarily as a result of design decisions taken with the client handset application. The intrinsic benefit of using TCP is that the connection-oriented transfer process aligns itself with the reliability requirements of the mobile sector. Communication is maintained because each telephone client retains contact with its respective network NAT / Firewall whilst these in turn communicate through the independent SysProxy server that is aware of both. Further design details of this solution are presented within chapter 4, notably sections 4.2 and 4.4.6.

3.4 Summary

This chapter has introduced and elaborated upon the key requirements necessary for the selection of an appropriate P2P routing algorithm. Highlighting the importance of selecting the correct technical solution, the chapter justified the progressive exclusion of numerous P2P overlay technologies to present five as candidates for further analysis: Chord, Pastry, Tapestry, CAN and Kademlia. The case was presented that all of these technologies could be capable of being implemented within a mobile domain primarily because of their similarity resulting in no significant operational advantage. Due to these circumstances, Kademlia was chosen in view of existing implementation evidence and experience presented within other academic work [53, 8, 28]. Furthermore, the chapter presented the problems caused by Network Address Translators in interrupting communication using interactive protocols such as those employed in P2P overlay networks. Descriptions of the generic solutions available to overcome these problems were discussed. These motivated the explanation of the proprietary hybrid solution implemented within the mobile framework discussed in the following chapters.

Chapter 4

Development & Implementation of a P2P Framework

The development of a P2P mobile framework commenced by evaluating a 'proof of concept' software application developed in 2008 where Kademlia was the underlying P2P transport mechanism [53, 8, 28]. This 'proof of concept' application, introduced in section 3.2.3, is referred to as the 'BUTE' (Budapest University of Technology & Economics) version to distinguish it from the enhanced 'Bushfire' version primarily presented within this thesis.

Utilising the BUTE software gave an outline framework from which to commence an evaluation and perform benchmarking of a P2P node application on a mobile device. Indications from online and printed literature were positive that the software could form the basis of future research associated with the Bushfire Framework. However, further analysis for the Bushfire project indicated the BUTE application did have several fundamental shortcomings (refer to section 4.2) which needed to be addressed if the Bushfire framework was to evolve from it. This chapter discusses elements of the BUTE application design as well as elaborating upon the architecture, design alterations and work performed for the Bushfire Framework implementation. First, however, section 4.1 outlines the principal design requirements for the Bushfire Framework in order for it to be considered a viable solution.

4.1 Design Requirements for the Bushfire Mobile Network P2P Framework

Different to a standard application, the objectives and requirements for designing a framework are drawn from primarily the software developer's perspective. Creating a P2P framework is no different to other frameworks developed for different situations; a combination of general and scenario specific requirements will exist. Conceiving a successful mobile P2P network framework must ensure the following requirements are considered:

i Overcome constraints when using the mobile telephone data networks. For example, ensure that mandatory functionality for modern networking environments, such as NAT traversal, is incorporated seamlessly and without burden for the programmer.

ii Overcome issues concerning variable node network connectivity. Even in areas with good mobile telephone network coverage, temporary loss of the data connection (and therefore suspension of P2P network operation) can occur through normal telephone call operation.

iii Speed up development time of P2P networked applications through an intuitive programming interface. This API should also help reduce operating / maintenance costs of P2P networked applications by allowing use of prior developed and tested code. A framework should aid, not hinder, the application development process.

iv Flexibly permit an alternative mobile application development model yet provide enough support to aid the construction process.

These technological framework requirements have been superimposed upon those already implicit by virtue of using of a P2P overlay on a commercial project. Chapter 3 has discussed P2P networks in general technical terms and highlighted the commercial constraints under which the project was operating. These design requirements above are cognisant of meeting these aims and set with appropriate scope to ensure success against research (list items i and ii) and commercial criteria (list items iii and iv).

4.2 From the BUTE to Bushfire Application - Summarising the Major Enhancements for a Mobile Network P2P Framework

As the BUTE application authors highlight within their documentation, Kademlia is an abstract design description and does not define the implementation details such as message channels or protocols. Therefore a number of the design decisions made by the authors were self imposed, being designed to meet the practical requirements of Kademlia and their perceived application requirements. These choices as well as the architecture were reviewed as part of the prototype evaluation exercise. For example, although the Kademlia P2P transport mechanism was utilised, the node client software was purposefully constructed to support an application specific scenario. This determined not only the data format / communication protocol but to some extent the general behaviour of the nodes.

The Bushfire Framework rose out of the P2P architecture foundations laid by the BUTE application. The design evaluation exercise indicated the BUTE application partially met the requirements for the Bushfire Framework (section 4.1). As amenable licensing conditions permitted access and modification of the BUTE application source code, adaptation rather than a new creation was deemed a viable approach. Moreover, the quality and design modularity of the existing BUTE source code was judged suitable further supporting this development avenue.

One substantial limitation of the BUTE application was a stipulation to operate on a handset capable of connecting to the internet through a wireless LAN (WLAN) connection. This requirement stemmed from a fundamental networking limitation discussed in section 3.1.1, namely no NAT traversal capability. Using a WLAN connection circumvents this problem if the nodes connected are within the same network address space or have a publicly visible IP address. For the purposes of proving hardware capabilities, software development and testing (as indicated in section 3.2.3) this approach was adequate. However, for 'real world' use case scenarios it is realistic to assume that a user will connect through the internet connection provided by their network operator and in different network address spaces.

A summary of the principal modifications required to correct shortcomings with the BUTE application, as well as the implemented extensions and enhancements, is outlined as follows:

i Removal of integrated application-specific source code from the BUTE code base to enable the introduction of a clear API, thereby ensuring the Bushfire Framework design is application independent.

ii Modification of data packet structure to support the new P2P framework API.

iii Connection of the P2P node onto the public GSM mobile telephone network (unlike the 'WiFi only' BUTE version) using relevant handset API functionality.

iv Integration of additional functionality into the framework system to permit full P2P overlay operation across a mobile telephone network. This was achieved through the introduction of 'Sysproxy' as a proprietary 'Proxy Server' application to support necessary NAT traversal.

As item i. implies, the BUTE application was not generic. Written to target a specific P2P file sharing protocol, the application was however constructed to permit code reuse. Sections 4.3 onwards elaborate further on applicable details. Whilst the Kademlia P2P overlay was functional in a basic form, item ii. illustrates the requirement to simplify and generalise the data packet structure to facilitate transport around the nodes in the Kademlia P2P overlay. Although operating the BUTE application on telephone handsets proved viability from the perspectives of computational processing power and memory usage, a major omission remained: network connectivity via a standard GSM mobile telephone network. Items iii. and iv. were necessary modifications as the BUTE application worked only using an 802.11 type (or 'WiFi') WLAN connection. Therefore mobile operation previously reported had only been accomplished with telephone handsets equipped with WiFi capability, working in range of a WLAN access point. Furthermore there was no NAT traversal capability within the BUTE application, resulting in P2P nodes only being able to connect to others with a local, private IP addresses. Whilst acceptable for experimentation, this configuration would obviously not be viable for 'real world' mobile operation. Utilising knowledge of NAT (section 3.3) and operating methods of mobile telephone networks, Bushfire achieved full internet and overlay connectivity through the operator network by the introduction of a proprietary 'proxy server'. Depending upon the criticality of the application environment, running the SysProxy proxy server within a hosted service internet environment might be appropriate. Section 4.4.6 elaborates on this fundamental addition to the framework system architecture. This was a design alteration to BUTE to permit P2P network activity in a truly mobile environment.

4.3 Architectural Design Description

A P2P network constructed using the BUTE application had the software for each peer node located solely upon the handset. However, as indicated in section 3.3.4, to overcome the constraints introduced by NAT and the mobile telephone network an external 'Proxy Server' application (code named 'SysProxy') was introduced into the system. Figure 4.1 illustrates the outline network topology of the Bushfire Framework with the SysProxy application at the core of communication paths between nodes (handsets) on the P2P network. The Kademlia P2P network is an overlay on top of a star topology network connectivity design.

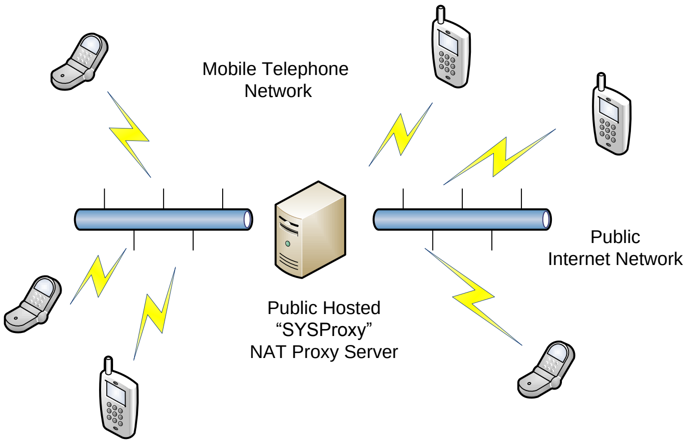


Figure 4.1: Outline Network Topology of a P2P Network using the Bushfire Framework

Figure 4.1 illustrates that the Bushfire P2P Mobile Network Framework comprises two interdependent software components: the handset and proxy server. Each element presents a different set of requirements yet both have a common goal of P2P network connectivity for the participating nodes. An alternative view of the system is presented in Figure 4.2. This block diagram illustrates more concisely the system level interaction between the constituent parts of the Bushfire Framework. The blocks relating to NAT traversal and API function relate to some of the additional functionality contained within the Bushfire system compared to the BUTE application.

Having a P2P network with a central proxy server could be interpreted as not compliant with the definition of this category of network. It is acknowledged that this configuration does have the significant weakness of a 'single point of failure' (i.e. no inter-device communication occurs when the SysProxy proxy server application is not operating). However, the behaviour of the nodes communicating with each other through the proxy application is fully compatible with P2P overlay operating techniques. Therefore the benefits of a distributed storage network remain a reality in the 'real world' (Bushfire) implementation. Depending upon the criticality of the application environment, running the SysProxy proxy server within a hosted service internet environment might be appropriate. Further discussion on how the design weakness of a single proxy server can be overcome takes place within

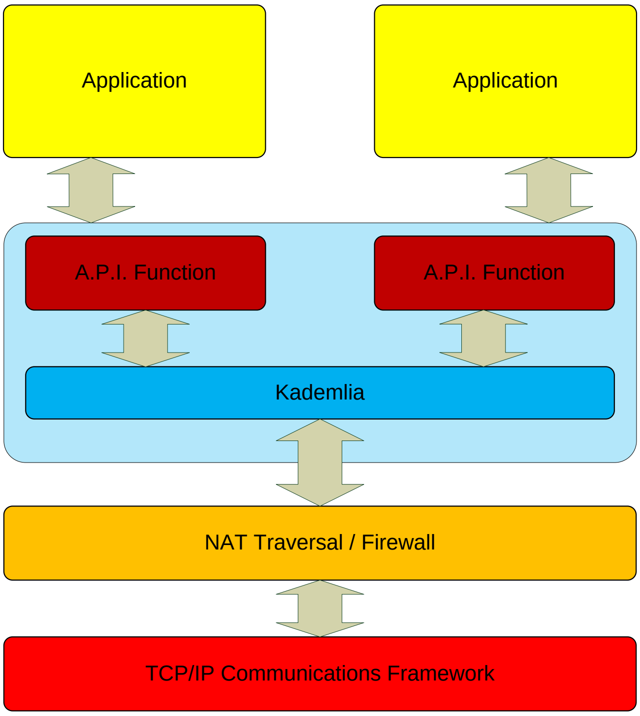


Figure 4.2: System Block Diagram for Bushfire Framework [7] (Adapted)

section 4.4.6.

One design goal of all the software used within the Bushfire Framework is for modularity of the software components. The intention is to promote ease of integration, design and maintenance through this approach. Figure 4.3 illustrates the basic system building blocks for the Bushfire Framework on the handset and how they relate to each other. The handset component, although implemented separately, is considered to include the applications as this contains necessary interfacing source code.

The design architecture of the Bushfire framework was also heavily influenced by practical implementation constraints from the BUTE application. Dependencies associated with using the Nokia Symbian S60 smartphone platform (as introduced at the start of chapter 3) were present and not unusual from a mobile telephone software development perspective. The application was implemented in native C and C++ code running on the Symbian Operating System (OS)(Version 9.1 & 9.2). Although Symbian is an OS that has been ported to several manufacturer handset platforms (e.g. Nokia, Samsung, LG, Sony-Ericsson) [161], there remain platform-specific configuration issues that dictate hardware choices. One such configuration issue was the availability of the general C library (glib), currently only on the Nokia platform variant. Another issue was the User Interface (UI) software framework that ran in unison with the Symbian OS. The BUTE / Bushfire applications utilised Nokia's S60 3rd Edition [158], heavily influenced by the previous library restriction. These two constraints affected the choice of handsets, limiting software operation to Nokia's S60 series of smartphone devices. Although specifying a device hardware platform could have been considered a limiting factor, the practical reality of this design choice had little impact upon viability and adoption due to the very large numbers of handsets in circulation [29].

The choice of handset hardware platform, which thereby dictated operating system and development languages, was a commercially driven requirement for this project. The availability of the BUTE application assisted achieving this goal and as a consequence also improved the delivery schedule. As stated in sections 4.2 and 4.3, the Kademlia overlay design is independent of implementation criteria such as processor or operating system. Therefore the framework architecture could be transferred to any capable smartphone platform and constructed using

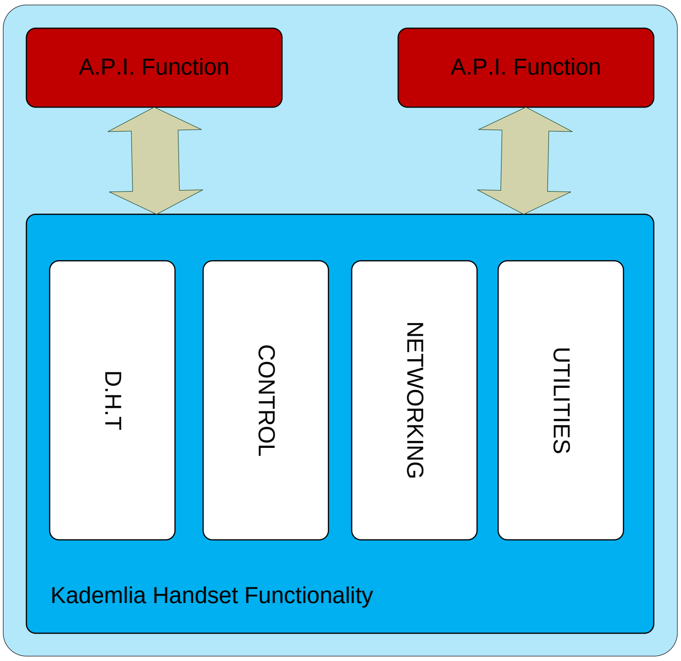


Figure 4.3: System Block Diagram for Bushfire Framework: Handset Component

the appropriate programming language (e.g. Java for Google Android, Objective C for Apple iOS). This approach requires a clear detailed design specification to ensure equivalent functionality is implemented upon each platform. In the case of the Nokia S60 platform implementation, the modular construction of the Bushfire framework (as discussed within section 4.4) supports its validation against the generated Kademlia overlay application requirements specification.

There are two distinct advantages for ensuring the overlay framework design is impartial to handset selection, specifically wider public adoption and avoiding obsolescence. With a variety of handset operating systems available to the consumer in addition to Nokia's Symbian (such as Google Android, Apple iOS or Blackberry OS), truly wide scale adoption of a P2P framework can realistically only occur when several are supported. The framework design paradigm and programming to an API methodology both encourage platform portability by clearly defining the framework and application scope. From a commercial perspective, these appeal to application programmers as they reduce the volume of source code required for different operating environments. Nokia's public announcement in September 2011 [14] indicating they were ceasing active development of its S60 platform smartphone platform by permanently outsourcing activities has provided evidence that accommodating platform obsolescence is a real requirement. Although the numbers of deployed handsets will remain large for the immediate future, given the typically short lifetime of consumer products, porting of the Bushfire framework onto another hardware / OS platform is inevitable for future commercial viability.

4.4 Detailed Design Description

Analysis of the design documentation and software code revealed the design architecture of the BUTE software as modular in part. Partially arranged to segment the application layer from the underlying Kademlia P2P transport layer, it was possible to determine which modules were suitable for inclusion within any future P2P client. The BUTE application functionality was divided into four distinct categories. The clear boundaries of these functional categories translated to a set of libraries within the software (named in italics).

- Networking functions (i.e. TCP/IP UDP communication) *[KiNetwork]*

- Kademlia Overlay Control / Processing *[KademliaCore, DHT]*
- Application Target Functionality *[BUTE application specific libraries removed.]*
- Test functions (e.g. logging)*[KiLogger, DHTGuiTest]*

Acting as a construction frame, some classes and libraries from the BUTE application remained in the enhanced code base to form the foundations upon which the Bushfire framework was built. Significant modification of numerous sections of the code base was necessary for the Bushfire application in order to accommodate fundamental changes like the data packet structure (discussed further in section 4.4.2 and Figure 4.4). The BUTE application-specific libraries were not included within the future development work and therefore related software code was extracted to leave appropriate communication interfaces. Test functionality remained (with necessary modification) within the updated design configuration as the requirement for logging and a test application remained appropriate. In a similar manner to installation of software upon a desktop PC, mobile handset software is installed upon the handset platform using the Nokia proprietary installation process. The above functions were integrated into four separate installation files and installed to a handset along with other pre-compiled libraries such as the Symbian 'glib' C library. An outline description of these principal libraries and functions follows.

4.4.1 Design Description: Networking Function

The KiNetwork library handles tasks associated with internet networking and the P2P overlays (and not functions associated with the telephony network). For example, functionality associated with implementing TCP/IP sockets was handled within this library. This included 'Reader' and 'Writer' operations to handle reception and transmission of data from the node to the network as well as timers to control / oversee this.

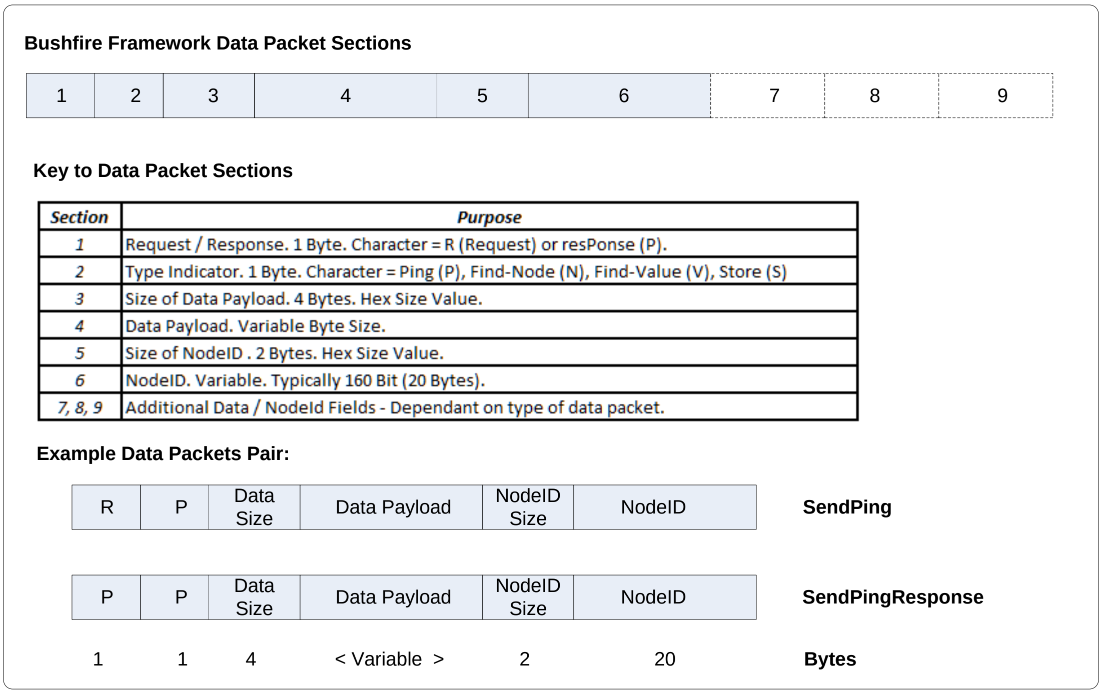
Enhancements to the KiNetwork library for the Bushfire framework included important functions associated with the node's interaction with the SysProxy server. For example, buffering of data to allow asynchronous operation with the SysProxy server ensured the sporadic, burst style transmission of data across the P2P network was catered for. The improved library also implemented additional functionality for retrieving the IMEI number from the handset. The 15 digit IMEI number is used to uniquely identify the device and is found upon almost all mobile telephones including all those which are GSM and WCDMA network compatible. Unlike the telephone number, which is assigned to a telephone via the operator network, the IMEI number is permanently programmed within a device and can be retrieved via API functionality defined at Symbian OS level. The unique properties of this number made it ideally suited for the node identification role upon the P2P overlay network. Utilising the IMEI code within Bushfire replaced an inferior integral node identifier function based upon quasi-random number generation. Given the anticipated number of nodes in the P2P network, the IMEI number solution appropriately meets the requirement of unique node identification.

4.4.2 Design Description: Kademlia P2P Processing

KademliaCore library is best described as a generic Kademlia implementation. As recognised by the software authors, the seminal Kademlia work does not give direction on implementation, only functional behaviour. The Kademlia library contained abstract interfaces, some requiring compulsory implementations, others optional, in order to create the Kademlia functionality required for the node. This design decision of mixed interfaces gave the most flexibility to the programmer in the implementation of necessary functions. Design choices were also taken relating to high level concepts (e.g. relationship between keys and values) and low level implementation detail (e.g. data packet design) for the framework.

With respect to data storage, the KademliaCore Library in its original configuration stored only one data item per key. Modification of the Bushfire framework did extend this functionality to allow multiple data items, but for simplicity of operation the configuration remained as a single 'key' related to a single 'data item' for the applications discussed in chapter 5. Despite limiting the amount of data associated with a key (or search) term (inconvenient for popular terms), the single associated data item approach aids retrieval reliability.

The construction of the data packets sent across the Bushfire framework was significantly re-designed from the BUTE application format to take into account a generic API. The data packet was designed to be compact, logical and ‘human interpretable’ to both minimise data traffic and processing overhead. Having a format that could be easily interpreted aided testing phases of the framework design as it was possible to inspect network packets by sight using software tools such as ‘WireShark’ [125]. Figure 4.4 illustrates pictorially the construction of this data packet. The total packet length was designed to be variable, comprising both fixed and variable length elements. For example the first and second bytes are permanently configured to indicate the packet purpose and type respectively. The first byte is a single character - ‘R’ (for Request) or ‘P’ (for resPonse). The second byte contains a character relating to the commands discussed below. Both bytes dictate how the subsequent data in the packet is interpreted by the node. Variable length sections of the packet are targeted towards payload data and consist of a size / data pairing (with size in the ’human interpretable’ format). This gives the packet size flexibility within the framework which allows both simple and complex application payload data to be conveyed as well as maintenance traffic.



Section	Purpose
1	Request / Response. 1 Byte. Character = R (Request) or resPonse (P).
2	Type Indicator. 1 Byte. Character = Ping (P), Find-Node (N), Find-Value (V), Store (S)
3	Size of Data Payload. 4 Bytes. Hex Size Value.
4	Data Payload. Variable Byte Size.
5	Size of NodeID . 2 Bytes. Hex Size Value.
6	NodeID. Variable. Typically 160 Bit (20 Bytes).
7, 8, 9	Additional Data / NodeId Fields - Dependant on type of data packet.

Figure 4.4: Example of Bushfire Framework Data Packet Construction Design

The Kademlia P2P overlay processing logic was incorporated into the libraries KademliaCore and SysDHT. SysDHT is an optimised Bushfire alternative implementation to the BUTE-specific library. These libraries include all code associated with the routing table and DHT. As well as DHT implementation they also cover all aspects of their behaviour such as establishment and maintenance. Additionally logic associated with related activities of finding and storing data into the network was implemented in these libraries, although specifics relating to networking functionality was not. The KademliaCore library also participates in routing table updates by tracking node messages that are dispatched to the network. The routing table maintains a correlation between nodeId and IP address. Whether a response is received or not, an update procedure is automatically triggered. Unresponsive nodes are removed from the routing table after a predetermined period in order to maintain a ‘fresh’ routing table state. Within the BUTE software code, the nodeId was a 20 bit random number. However this functionality has been superseded as the generation process was not considered robust enough to generate non-repeating node identifiers. Section 4.4.1 introduced the use of the IMEI number as a viable and appropriate alternative, so this approach became the new source for the nodeId. Using an IMEI based nodeId was also progression in terms of making the system more secure. Although IMEI numbers on their own could still be copied and used maliciously, if coupled in a future design iteration with other similar user unique, known data (such as telephone number or

SIM serial number) then a method to prevent 'nodeId spoofing' could be implemented.

As defined in the original BUTE implementation, all of the high level logical activities associated with routing, DHT maintenance and data storage / retrieval from the P2P overlay network are handled by the implementation of four simplified commands (refer to following list). Commands iii and iv are more obviously related to the overall objectives of the Bushfire framework of storing data and retrieving it from the network. However, to achieve this functionality, commands i and ii are a composite part of these with necessary additional logic.

i **ping:** Command to test for the presence of a node within the P2P overlay network. A node will either send or respond to this command in order to determine current overlay connection status. Ping is the simplest command within the group and is a component part of the other functions.

ii **findNode:** Command to locate a particular node within the overlay network, performed using a sequence of Ping commands integrated with logic to monitor responses received from queried nodes. findNode is used to locate a specific node on the network, for example a node that holds the desired data itself or knows of more detailed routing information.

iii **findValue:** Command to retrieve data from the P2P overlay network. This command comprises a findNode function call and the logic to return the desired data value if located on queried node.

iv **store:** Command issued to store data to a particular node within the network. This command comprises a findNode function call and the logic to store the required value on the closest applicable node currently present within the network.

Building upon this, each Kademlia function has a pair of software methods relating to the communications path associated with it. These are named simply as sendXXX and receiveXXX, where XXX is any one of the commands described above. The correct function to be used is determined by whether the node is acting as a client (receiver) or server (transmitter) role in the current P2P data exchange. For example, there are four functions associated with the Ping command:

- **sendPing:** Command for 'client role' node to send a message to test if a particular node is on the network.
- **receivePing:** Command initiated at a 'server role' node upon receipt of a ping message from another node upon the network. This command processes the received message (including updating its own routing table) and then initiates a response.
- **sendPingResponse:** Command for a 'server role' node to initiate a ping response to the node which sent the original inbound ping message. This command is reactionary and is called by the receivePing command.
- **receivePingResponse:** Reactionary command for a 'client role' node to instigate upon receipt of a response message sent as a result of its initial outbound ping message.

These four commands are displayed in the order in which they would be called and processed as a 'conversation' between two nodes within a P2P overlay. The 'client role' node is at the start and end of the message exchange, with the 'server role' node accepting the ping request and responding accordingly. Figure 4.5 illustrates the timing sequence of these function calls for an example scenario of finding a node. A node is able to issue multiple sendPing requests as well as handle the asynchronous responses. The 'routing table refresh procedure' uses this procedure to determine node availability status. To achieve multiple requests, a node creates an array of queued requests, each with a unique transaction identifier. An 'observer' software model is implemented to asynchronously invoke the appropriate response routine when a reply to a request is received. To maintain correct operation, a 'time out' timer is used to delete any unanswered requests from the node, which typically relates to another node that has left the overlay. Additionally, to control responsiveness of the node, the number of parallel requests in progress at any one time is configurable. The default of these control parameters is 10 seconds and 64 active calls respectively but could be subject to variation following future experimentation.

From the commands and sequences discussed above, it is clear that the operation of all the Kademlia functions had been designed to utilise combinations of simple programming functions to construct more complicated tasks. This component design approach ensured maximum code re-usability whilst aiding clearer, more concise code construction. For example, the Kademlia FindNode functionality was implemented using multiple Ping commands with additional processing logic to coordinate the multiple calls and responses.

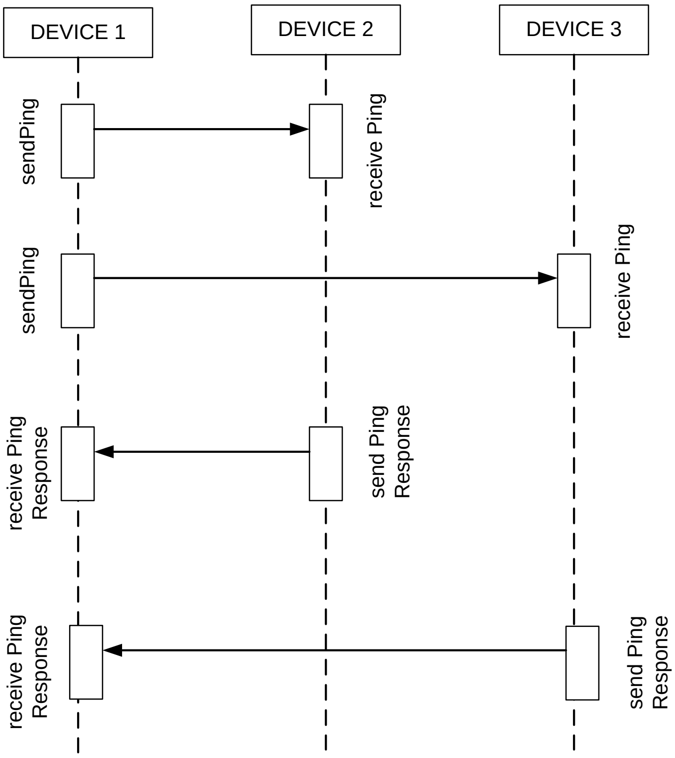


Figure 4.5: Example Sequence Diagram of Function Call

4.4.3 Design Description: Application Function

With reference to the original BUTE application [8, 28], software libraries associated with this functionality were removed as no longer applicable to the framework construction. Applications would instead integrate with the Bushfire framework via a new API. Software frameworks and APIs are created for the primary purpose of allowing easier access to and closer integration of third party applications with a particular host application. This clear separation of scope and responsibility aids assuring greater application reliability and robustness. For the Bushfire framework, the objective is to permit applications to integrate into a P2P network. This abstraction of the application away from the underlying P2P data transport layer simplifies the development process in terms of functionality scope reduction and testing requirements.

The BUTE software code base provided an interface for an application via the 'DHTInterface' class. However from inspection this was considered too closely coupled with the previous application scenario (and thus no longer applicable to the Bushfire framework development). Therefore this original code was removed and improvements put in place to make its replacement less coupled to the application and easier to understand. The code that implements the API concept within Bushfire is accessible through a 'version 2' of this class (also named 'DHTInterface') and other supporting code. However the design now conceptually consists of two sections: part located within the framework code base, part integrated within the application. These two sections are at the core of communication between the framework and the application and are discussed further within chapter 5. This class brings together access to all applicable functions from the various classes and libraries in the framework for use by the end-user third party application. The following list illustrates an example selection of API functions available to the application. Functions with overtly descriptive names exist within the 'DHTInterface' class to aid developers in selecting the most appropriate commands to utilise.

i **StartKademliaNetworkConnection:** Initiates a connection to the other Kademlia nodes once the telephone network and SysProxy connections have been established. Requiring no arguments, this function starts the DHT applicable code running and from its contents determines if enough routing data exists to join an existing peer network. If this is not the case (as for a newly registered user) a function is called to obtain the address (Port number) of a known node on the network (typically the user who sent the application invitation).

ii **UploadDataToNetwork:** Function to oversee uploading a 'Key Value' pair of data into the network. Arguments for this function consist of general data buffers containing the predetermined key data to be stored along with the value data. This function performs several compatibility, error checking and connection tests before the data is assigned to the overlay network using the lower level function 'PutValueIntoNetwork'.

iii **DownloadDataFromNetwork:** Function to oversee retrieving of data from the network based upon a search term (key). This function uses a buffer argument to input the search key term and return the appropriate value data. As with the function 'UploadDataToNetwork', simple checks are made to avoid error conditions crashing the application such as checking for a valid overlay connection before instigating the query.

iv **BuildDHT:** Command to initialise the construction of the DHT within the network node. This is a top level command with no arguments, causing lower level functions such as 'BuildSysDHT' to load the data for the node DHT from permanent file storage on the handset.

v **ResetNetworkNode:** Used to instigate a manual reset of the network node.

API commands are broadly categorised into two sets, basic and detailed. The API functions belonging to the basic command set (such as i, ii & iii) were of a higher level of abstraction and more appropriate for general control requirements of the node. These commands illustrate that storage and retrieval of data, once connected to the P2P network, can be performed via a straightforward interface. Conversely API functions belonging to the detailed command set (such as iv & v) were involved with finer control elements of the node behaviour. For example the 'BuildDHT' function instigates the construction of the local node DHT. This functionality could be controlled by either the program specifically or 'wrapped up' within a higher level task to give more autonomous operation.

An example of where a combination of API commands are used is the join sequence for a node to connect to the Bushfire framework. The connection of a node into the Kademlia overlay is conceptually easy to comprehend at a high level of abstraction. The detail of implementation reveals a number of discrete stages to be processed before a node can be considered as an active overlay participant. Definition of this could be taken as communication where routing table data is exchanged between nodes with the purpose of supporting accurate target data acquisition. Within the Bushfire Framework, there is the process of connecting the handset to the SysProxy host machine as well as the node to the P2P overlay network. The SysProxy connection is discussed more thoroughly within section 4.4.6. Figure 4.6 however illustrates the overlay join procedure for a node using a state diagram. The numerous state transitions/function calls that need to be made are summarised.

The join sequence illustrated consists of contacting a number of nodes within the routing table, up to a limit set by the number of maximum simultaneous node searches. The contacting process is handled by a TaskManager class charged with overseeing each batch of tasks associated with a node operation. In the case of joining an overlay, the TaskManager calls the StartFindNode function which instigates the functions at the start of the node finding sequence. This is depicted as the lightly shaded green section in Figure 4.6. Function FindKClosestNodes dictates how many nodes (K) are going to be searched for simultaneously on the overlay. This needs to be defined because multiple search tasks can be set going together by the function FindNodeTask. When defining the value K, a compromise is sought between the speed of updating a routing table and the volume of maintenance data traffic on the overlay. When a search task is started, a message to query nodes is sent onto the overlay and the observer function started. There is an array of tasks, each task operating in its own thread. The purpose of each task is to find the node it has been assigned from the routing table. Finding at least one other active node in the overlay from routing data constitutes a successful join status. The result of this node search is received back asynchronously at the node and the appropriate observer is awoken. The resulting operations called by the observer class, such as updating the routing table and deleting the tasks from the array, are illustrated in the darker shaded red section of Figure 4.6. All task management and routing table maintenance operates using timers within the software. Time activated operations ensure certain essential tasks are performed regularly throughout a node's autonomous participation in the overlay. An example of a timer is the Bucket Refresh Timer, used to ensure the validity of the contents of the routing table.

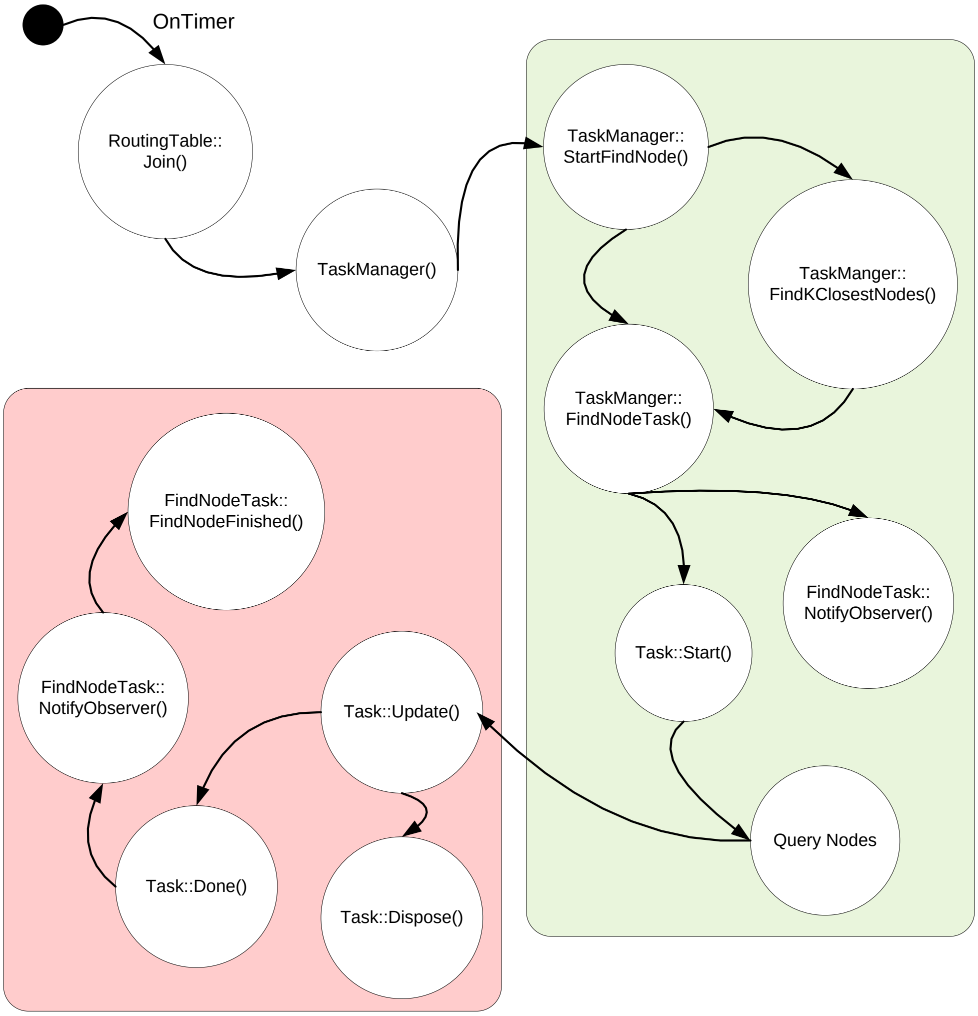


Figure 4.6: Quasi-State Diagram of Overlay Network Join Process

4.4.4 Design Description: Test and Debug Related Function

Functionality associated with testing and debugging existed within the original BUTE software code base. Recognised as important in the construction and maintenance of software, the functionality remained within the application primarily for the formative stages of the framework development. The class ‘KiLogger’ implemented the logging function, writing text data to a log file stored upon the handset. This functionality was indispensable for ‘post run’ analysis of the application when ascertaining ‘code run progress’ of an application before a crash point. Once stable, the log function also provided a means to record application ‘health check’ status. This is conducted by writing ‘progress markers’ to the log file as node operation proceeded through different sections of the code. This process ensured that when new functionality is added to the framework no negative effect is introduced into existing code. ‘DHTGUITest’ is a class used to implement some of the test functionality which the logging class was monitoring. This class provided the forerunner idea to a full test application that was implemented for Bushfire framework testing known as ‘BushfireTestEngine’. This application is discussed more fully within section 5.1.

4.4.5 Design Description: Handset Component

As depicted in Figure 4.1, the system as a whole comprises two functional (also physical) components: the handset and the SysProxy server. Elaborating upon the content of section 4.3, the handset component of the Bushfire framework comprises the core framework code and the user application logically above it in the software stack (Figure 4.3). Therefore handset functionality is associated with all aspects of the system: integration with the handset, interfacing with SysProxy and operation of a Kademlia P2P node.

Handset Component: Integration with the Handset

Growing a P2P network from a seed node is understood to be difficult. For Bushfire however this is different since users already have information on their own existing network of family, friends, work colleagues, etc. stored within their phone. Whilst handset design can vary considerably, the presence of key functionality such as phone book access and text messaging remains consistent. The Bushfire Framework utilises these functions when connecting to other people and communicating the P2P network's existence. It is envisaged that most users will learn of Bushfire Framework applications through an existing known contact. Handset integration features in the form of 'helper classes' supplementing the standard handset APIs have been created for the Bushfire Framework and made available to P2P applications for this purpose. These classes extrapolate the necessary handset functionality to a higher, more appropriate level of abstraction for the Bushfire Framework. Within the handset integration feature set, two significant helper classes are 'ContactsInterogator' and 'SMSMessageSender'. ContactsInterogator will allow the user to select, through the P2P application, a known contact's data (such as mobile telephone number) for a specific purpose. SMSMessageSender will utilise ContactsInterogator to enable an SMS message to be sent to a known entry in the handset phone book. An example use case for the SMSMessageSender is to construct a text message to send to a new user, containing an invitation and URL to download the application (and thereby grow the number of participants in the overlay network).

As described in section 4.4.1, the unique identification of a node on the P2P network utilises the IMEI retrieved from the handset. Another unique identifier more commonly known to users is their telephone number. When used in conjunction with the standard format international dial code (such as +44 for the United Kingdom), their telephone number becomes a unique (worldwide) identifier that is known to others. The telephone number can become the key associated with a value in a Key:Value pair partnership on the P2P network. Unfortunately this telephone number is not obtainable from the Handset API as it 'conceptually' exists only at the telephone network operator system level (and not on the handset hardware). Therefore this problem is circumvented by Framework applications prompting the user to enter their own telephone number as part of the start-up configuration procedure. The telephone number is also an example of user data that is stored on the handset file system and accessible to applications via the Bushfire API.

Handset Component: The Kademlia P2P System and SysProxy Interface

As section 4.3 states, the Bushfire P2P Framework comprises both the Kademlia P2P system and the SysProxy communications application. Their relationship is interwoven as there is no P2P network without nodes (handsets) or the supporting infrastructure (SysProxy, internet and telephone operator network) for them to communicate. Data interchange between nodes will pass through all these systems during every communication sequence. Moreover, this all occurs over a TCP/IP communications link on top of a GSM radio channel between handsets on the operator and internet networks. However, these extra 'wrappers' surrounding the data all ensure the connection is maintained at its assigned level of operability. SysProxy has been developed using standard internet (TCP/IP) connection protocols so that connections between the nodes remain consistent across networks. From the handset component perspective, a continuous link is presented between sender and recipient. Considering a reduced scope and a high level of abstraction, Figure 4.7 illustrates the interaction between two devices and the SysProxy application. An example scenario of searching for a data value on a particular node is presented. From an overlay message perspective, SysProxy simply forwards the command onto the appropriate device. With reference to section 4.4.2, commencing with a sendPing command to determine a node's presence; the findValue routine augments the functionality further. Assuming a valid response to sendPing, findValue instigates additional tasks to retrieve the required data from a target node.

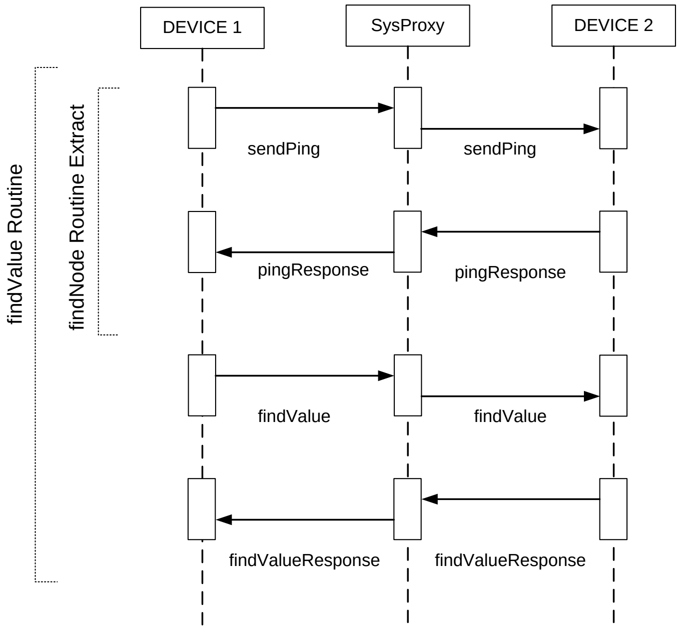


Figure 4.7: Sequence Diagram for Handset and SysProxy Interaction within Bushfire Framework

4.4.6 Design Description: SysProxy Component

Internet connectivity via a PSTN mobile telephone network connection is different to what typically exists on other network connections. With reference to section 3.1.1, mobile carrier connections work on the premise that all inbound data is the result of an outbound data request (with respect to the handset). The network connection has been specifically configured so that each handset is a network of just one device behind a NAT device. The Network Operators not allocating a static IP address to handsets is primarily for cost (due to scarcity) and security reasons (prevents direct connection by IP address masking) [47]. (Note that although some companies provide 'Fixed IP address SIMS' to the market place, these in fact use VPN technology to establish a virtual presence on a network that can be assigned a static IP address [48]. This does not represent a solution for the majority of standard mobile network users because of cost, availability and data consumption overhead.) This network connection behaviour in fact describes that of a Symmetric NAT that has been detailed previously (section 3.3.3). These constraints have severe implications for handset nodes in that traditional P2P communication (i.e. unconstrained inter-communication between nodes) is not permitted. NAT implementation decisions by the mobile network carriers are beyond choice/influence of their subscribers, thus it heavily constrained and influenced the implemented solution for the Bushfire Framework. Overcoming this restrictive NAT behaviour problem resulted in the design and integration of the SysProxy server into the core of the P2P network as introduced in section 4.3. SysProxy is an application intended as an essential conduit for all P2P message communication between participating handset nodes. It supports the overlay's existence (by routing messages between nodes) but does not participate in a manner representative of a peer node.

Vital to the operation of SysProxy is that, unlike all handset nodes on an operator's network, it is not positioned behind a NAT device (i.e. it possesses a 'visible' public IP address). This is a necessary constraint because all mobile devices need to be aware of the machine IP address they will connect to upon start up of their P2P node software. From the mobile network operator's perspective, the handset only has one data connection (to SysProxy) which it initiated. Mobile network operators are not concerned where the data passed across this connection is destined for or originates from, hence connections with other handsets are permitted. Figure 4.8 illustrates the intermediary operator firewalls / NAT devices present in the communications path between handset node and SysProxy. As is shown, with SysProxy using a public IP address, the system can operate with these typical constraints in place.

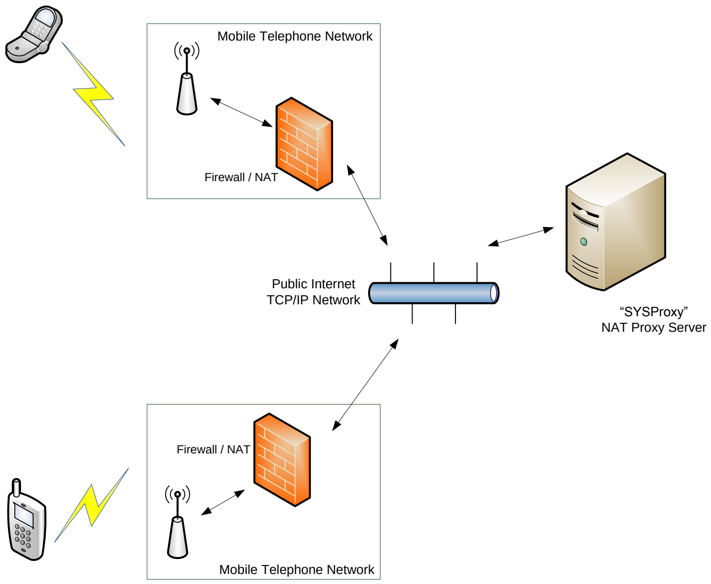


Figure 4.8: SysProxy NAT Proxy Server

Figure 4.9 illustrates the concept that SysProxy maintains a fixed connection with each participating handset (through a TCP Socket assignment) and then routes data to the intended device using the assigned port number. Although appearing to operate in a 'server style' role within the overlay network, the reality is that SysProxy does not store any user data from nodes participating in the P2P network. The internal operation of SysProxy, however, is constructed around the client/server model. An individual client/server relationship is established between SysProxy and every handset node it is communicating with. Data packet inspection processes are performed to allow routing of the data between nodes to occur. The client/server link concept is used at the 'handset connection' level because it is the model the network operators are accustomed to support (eg. such as web page requests). In brief, SysProxy receives data on one input socket and forwards it on to the appropriate output socket. This is illustrated simplistically within Figure 4.9. Nodes remain attached to a SysProxy port throughout their entire overlay connection time but messages delivered through the port will be intermittent depending on overlay activity.

Therefore in addition to fulfilling a role of 'gateway' to the P2P network for a mobile device, SysProxy also performs its own address mapping function. This is conducted by allocating a port number to each device that connects to the SysProxy and creating a cross reference to other P2P nodes on the network. A look-up table of handset IP addresses relating to SysProxy port numbers is produced and acts as the translation key between the physical network (IP address) and the logical overlay (Port number). The P2P overlay works by using routing table data to map Port number to NodeId. (Note that the routing table actually records the full UDP IP address (IP Octets & Port Number) but, given that all traffic is through the SysProxy, the IP Octets will be the same throughout with only a difference in the port number.) The physical port connection remains consistent and active whilst a node remains online and participating in the P2P network.

Maintaining a consistent port connection for a node is a function that has been especially implemented for the Bushfire framework to counter measures put in place by the mobile networks to close apparently dormant connections. For a typical client/server application interaction process (such as with a web browser) a TCP/IP connection is set up at the initial request and automatically torn down after a period of inactivity. However, connection activity of a node participating in a P2P network is more intermittent therefore posing a problem for maintaining a constant TCP/IP connection. Network activity for a particular node can be prompted by several reasons, such as a data request or routing table refresh activity, but they do not have to be instigated by the node user. As determining this network activity behaviour accurately is difficult in the P2P overlay, it was more logical to transfer the connection responsibility to SysProxy. This is achieved using a timed 'keep alive' handshake process (data exchange) to prevent the connection being dropped by the mobile network at a TCP/IP level.

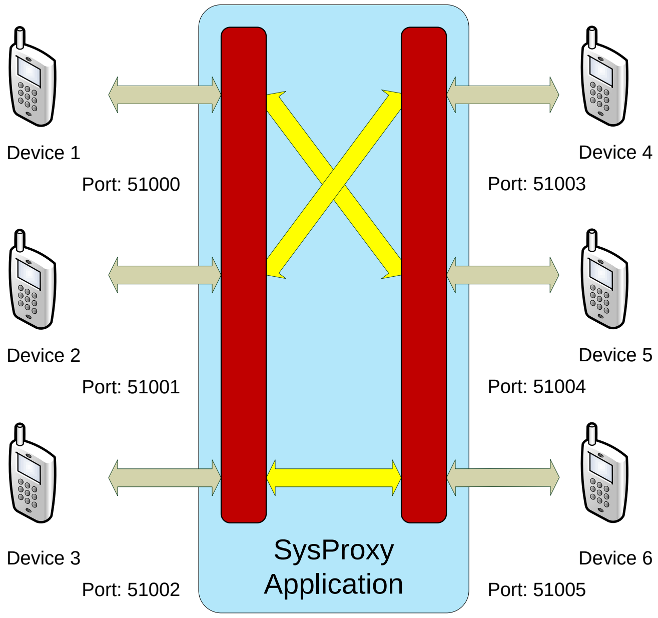


Figure 4.9: Operational Diagram for SysProxy

Figure 4.10 illustrates the simple repetitive systems structure of SysProxy making extensive use multi-threaded code to route data from one device node to another. SysProxy presents only a minor processing load whilst implementing ‘routing functionality’; the computational power demand is minimal and the storage capacity requirements remain consistently negligible over time (as no user data is stored).

As has been acknowledged previously in chapter 3, a mobile P2P network can be particularly susceptible to churn due to variability of carrier network connectivity. However, SysProxy has been designed to aid a reduction in this churn by minimising disruption caused by temporary operator network signal loss. (This is in addition to techniques implemented by the network operators to maintain a connection such as those employed during the call handover process between cell base station sites.) As SysProxy maintains the list of ports assigned to handsets, it will ’reserve’ indefinitely a port for a particular handset using its knowledge of the IMEI (NodeID) even after disconnection. After a network loss event the handset will attempt to re-establish connection with SysProxy on the same port. This port assignment technique also represents a design weakness as SysProxy is limited in the number of ports it can provide and therefore the number of nodes within the P2P network overlayed on top of it. The current design is crudely limited to 14535 nodes routing through a SysProxy application based upon the number of free TCP/IP ports available.

Correcting the weaknesses of the current SysProxy implementation, principally scalability and reliability, represent a design challenge for the future implementations of the Bushfire Framework. Both the restricted number of port connections and the ‘single point of failure’ limitation are inherent within the current configuration. Therefore one proposed solution to address these specific concerns is the introduction of multiple SysProxy nodes. Arranged in a logical hierarchical ring, the additional SysProxy nodes could be organised in a similar manner to the design described in section 2.4.3. SysProxy servers would exchange routing data information between themselves, as ‘super nodes’, related to the mobile nodes that are connected with each of them directly. From the mobile nodes’ network perspective, the operation would remain unchanged other than being able to communicate with a significantly larger peer group. Figure 4.11 illustrates the design concept. The topology of the network naturally aligns itself with using another P2P overlay for purely SysProxy information exchange. Different SysProxy server instantiations could then join and leave the ‘super node’ ring without a negative impact upon routing data exchange between mobile nodes connected to different servers. If the SysProxy network were to be segmented hierarchically, other implementation arrangements could be introduced to assist with the likely routing

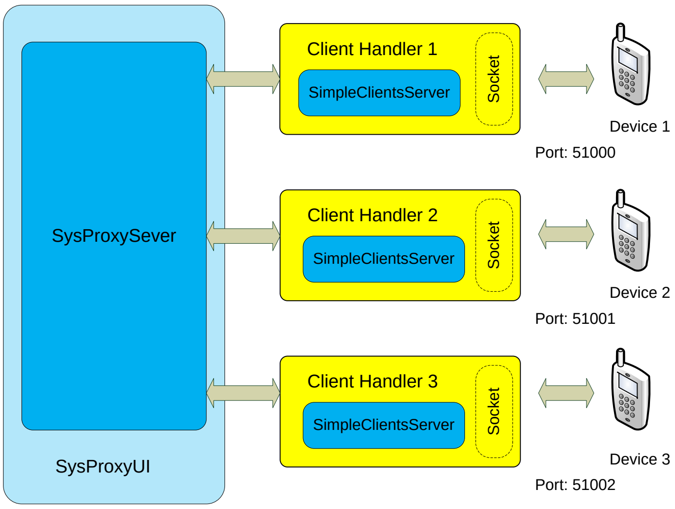


Figure 4.10: System Block Diagram of SysProxy

paths between mobile nodes. SysProxy servers could be arranged to minimise 'inter-SysProxy' communication and thereby minimise message latency.

4.4.7 Application Security Design Considerations

Security is an important aspect in all networks and network connected applications; P2P based systems are no different. Put simply, any time you let others access your computer (or telephone handset) you are potentially compromising on security [75]. When considering the creation of a P2P framework it is reasonable to assume that security aware design techniques and features are implemented as a matter of standard practice. In the case of this project, security concerns relate to both preventing unauthorised access to private data and communication reliability. It is the inherent, shared hardware architecture of P2P networks that present one obvious privacy concern relating to data storage. However, as P2P networks are typically large scale, any significant ability to prevent inter-node communication will have a disruptive effect on users. As acknowledged by Miller [75] and other texts (e.g. Buford et al. [65]), security concerns are complex and always evolving and therefore have 'no quick and easy solutions'. From the perspective of the Bushfire project, commercial time constraints of the initial research phase have limited the scope of considerations from practical implementation to design awareness only. Prior to any sizable public or commercial release of the Bushfire framework, implementation of specific security related functionality, such as data encryption or authentication mechanisms, would be necessary. As general users are considered a weakness from a security viewpoint [126], features need to be seamlessly integrated from their perspective with acceptable levels of inconvenience and risk (such as password protection). From the aspect of software development, functionality should either be fully transparent or accessible through the documented API. According to Saxena et al. [167], a proficient level of P2P security is based on the premise that the following three criteria are met:

- Secure communication channel between peers.
- Trust management or reputation systems to 'know your peers'.
- Access control to ensure secure admission onto the P2P network.

A secure communication channel can be implemented within the Bushfire P2P framework assuming the P2P overlay is administered securely. Secure administration would require some form of encryption to mask node identifiers, for the routing table data held upon a node and in the routing communication between nodes. Encryption of this data with a user derived key would prevent a rogue node from replicating and masquerading as legitimate through 'packet sniffing' or 'packet spoofing' techniques. A trust or reputation system can help significantly by obtaining part of the security authentication process from communication surrounding the logical,

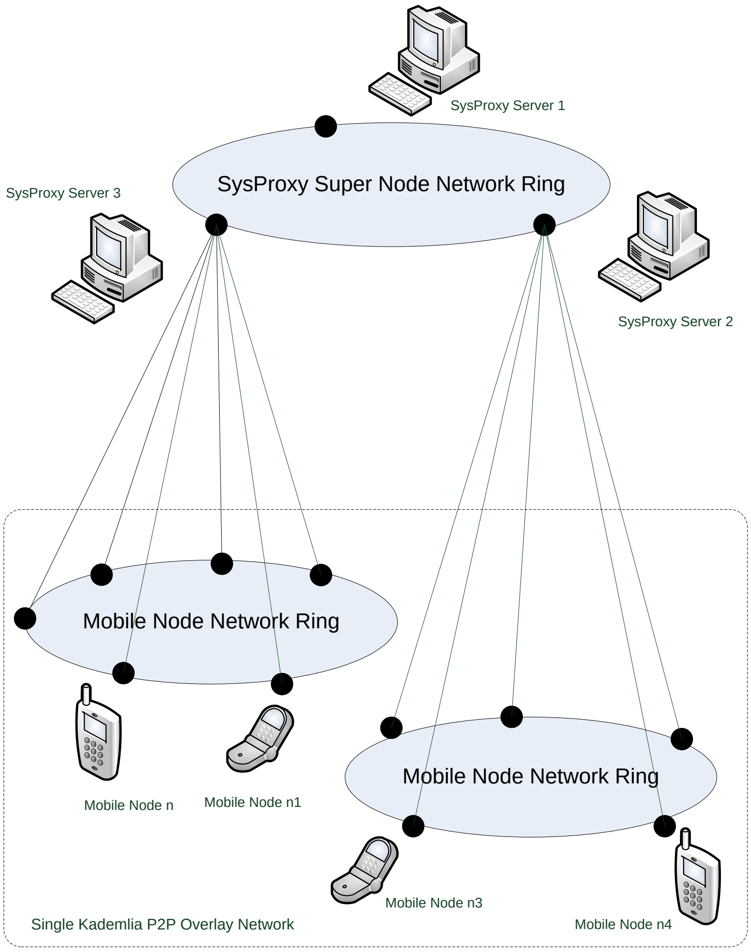


Figure 4.11: Multiple SysProxy Concept

abstract relationships between peers. These links may bring an implied trust relationship to the network connection. For example, a higher level of trust can be inferred for peers that belong to your family or work organisation than for a complete stranger. And finally, access control happens to some extent through limited physical access to the mobile device. However, further authentication maybe required such as user passwords. Within the Bushfire framework, the SysProxy server could provide access control services as currently all inter-node communication has to be routed through it.

Many academic papers have been written on this evolving topic since P2P networking became a mainstream accepted concept [132, 118, 126, 167, 21]. With reference to these papers, a summary of the principal, relevant security concerns appropriate to the Bushfire framework level of this project is given below:

- ‘Sybil attack’: A malicious application on a node forges its identity in order to easily access restricted data by ‘user masquerading’. This could be implemented by overriding the node authentication mechanism or data packet spoofing on the overlay.

- ‘Unauthorised data accessibility’: As user data is dispersed across the multiple nodes participating in the P2P overlay, adequate design diligence is required to ensure individual users cannot read, direct from their handset, private information belonging to other people. Encryption of framework level communication packets (incorporating user data) will reduce the risk of a data breach.

- ‘Distributed Denial of Service’ (DDOS) attack: DDOS attacks are targeted towards peers but also can cripple network overlay infrastructure by excessive request demands being made on particular node(s). The Bushfire framework design is particularly susceptible to a DDOS attack because of the single SysProxy server (which acts as a single point of failure). This attack could be rendered ineffective if the 'multiple SysProxy' solution as discussed in section 4.4.6 is implemented. The authors of the original Kademlia paper [71] also note that there is an inherent resilience to DDOS attacks by virtue of the routing behaviour. In short, routing tables cannot be flushed by flooding the overlay network with new peers as the routing table will only be updated when the old peers leave the network.

Section 5.4 discusses application security as a topic necessary for future consideration when developing software to be used in conjunction with the Bushfire framework.

4.5 Summary

With reference to section 4.1, it has been progressively shown across the chapter that the Bushfire P2P Framework has met all of the requirements proposed as necessary for a successful implementation. Building on the foundations of earlier investigations, development of the Bushfire Framework has been possible through application of knowledge obtained about the properties of mobile devices, networks and NAT behaviour. Implementing a solution that has combined the traditional aspects of a P2P network design with the alternative approach of a proxy server (SysProxy) at the centre of a 'star network configuration' has ensured interoperability is possible in the restrictive mobile telephone network environment. Critics of this design may argue that the single point of failure that SysProxy presents to this topology goes fundamentally against certain core traits of P2P networks, namely resilience, reliability and scalability. The counter response to this argument is that the benefits of the other properties of P2P network design such as dispersed data storage outweigh the disadvantages, particularly in a commercial environment. Supported further by the fact that the SysProxy application processing overhead demand is minimal (the majority of which is consumed by a Java Runtime Virtual Machine). Moreover, in any future commercial implementation, SysProxy could offer a network activity monitoring port for Bushfire Framework application developers or operators. Through this extended SysProxy application effective user billing and license enforcement functionality could be implemented (using data packet inspection techniques as all data passes through SysProxy). The commercial advantages of using an API are also well known and frequently exploited in terms of reducing programming effort required to get an application functioning. The intention to incorporate an API into Bushfire to enable other applications to utilise its P2P network node functionality was present from the beginning. Fundamentally, Bushfire is a framework, designed not to be an application operating on its own. The design, structure and functionality of applications integrating with the Bushfire framework have to be cognisant of data exchange amongst nodes on a network. The initial applications constructed for use with the Bushfire framework are discussed in chapter 5.

Chapter 5

P2P Framework Applications

With reference to section 4.4.3, software frameworks and APIs are created for the primary purpose of allowing closer integration of third party applications to a particular host application. The API of the Bushfire Framework provided a simple interface to a distributed storage system. This chapter introduces three applications that were developed to work with the Bushfire Framework, each using this API. All served a different purpose in the framework development process, from initial testing through to demonstrating commercial viability. The subsequent discussion draws out salient points of the applications and their integration with the framework.

Three applications, Bushfire Test Engine, BuddyNet and SupportNet, were all developed for integrating with the Bushfire Framework. All had an overarching requirement to store data within a distributed system. Equally applicable were the generic requirements that were established for commercial applications (as described within chapter 1 and section 3.1.2). In line with API design methodology, these applications could be written independently of the Bushfire framework with the exception of some interfacing method calls. Integration to the API is performed using a simple collection of functions. The commercial advantages for application developers using an API are well known: scope and effort reduction that in turn reduce the cost and time to market. Figure 5.1 illustrates the concept that applications will interact with the Bushfire Framework via the API. However, from the handset perspective all sections appear installed as a single unit.

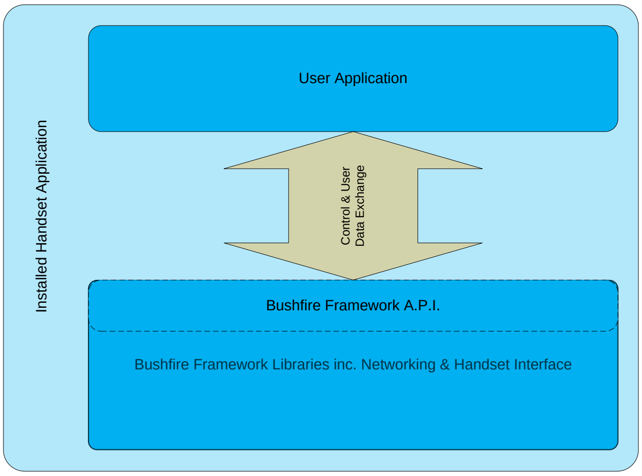

Installed Handset Application
User Application
Control & User Data Exchange
Bushfire Framework A.P.I.
Bushfire Framework Libraries inc. Networking & Handset Interface

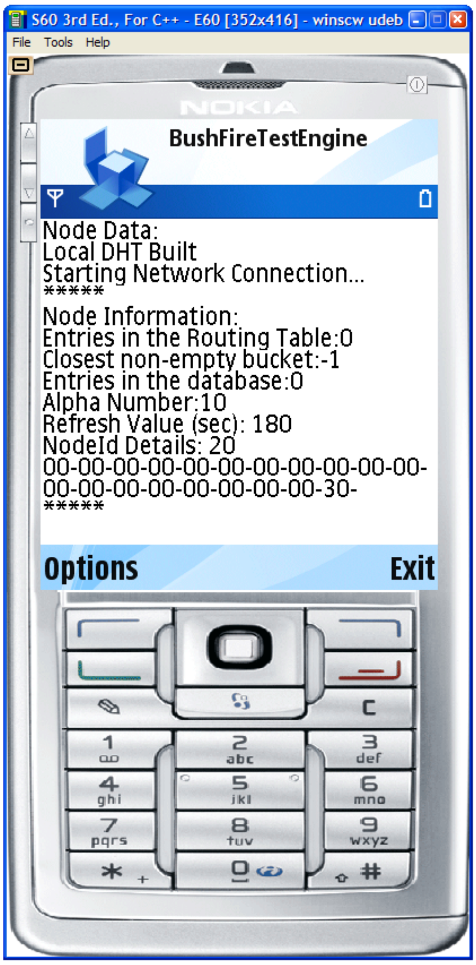
Figure 5.1: Functional diagram illustrating application scope in relation to Bushfire Framework API interface

Applications developed to work with the Bushfire Framework were created on an incremental basis. Verification of correct operation was performed through frequent operational testing. Initially, an application (Bushfire

Test Engine) with minimal functionality was developed to test the API for the Bushfire Framework. As the Framework functionality was extended and operational confidence grew, two more advanced applications (BuddyNet, SupportNet) were developed that were progressively closer to commercial grade software. Note that none of these applications could have worked with the BUTE version application discussed in chapter 4 due to differences in the APIs and the specific nature of the original design.

5.1 Handset Application: Bushfire Test Engine

'Bushfire Test Engine' is a framework test application only. Its initial purpose was to evaluate both the API interface and exercise the functionality associated with the Framework's implementation technology - the Kademlia P2P overlay. It achieved this through storage and retrieval of textual string data. To focus on the framework, the application was designed to be, in functional terms, as 'light weight' as possible. This also lowered the overhead of finding errors when new functionality was developed and integrated. The Bushfire Test Engine, although handset capable, was more frequently operated upon a handset PC emulator in unison with debugging tools (see Figure 5.2). This approach supported rapid application development through reduced deployment time of an emulator compared to a handset. An emulator also permitted more detailed analysis of the code during runtime, which is useful for embedded software. Figure 5.2 illustrates typical debug data presented to the user such as confirming 'start up' stages and configuration parameters. For example, the node identifier (the long sequence of digits set to ...00-30) and routing table data gives the operator valuable feedback on assessing the node behaviour in the network.



```
Node Data:
Local DHT Built
Starting Network Connection...
*****
Node Information:
Entries in the Routing Table:0
Closest non-empty bucket:-1
Entries in the database:0
Alpha Number:10
Refresh Value (sec): 180
NodeId Details: 20
00-00-00-00-00-00-00-00-00-00-00-
00-00-00-00-00-00-00-00-30-
*****
```

Figure 5.2: Nokia Software Development Tool ('Carbide') [31, 140]: S60 Handset Emulator running Bushfire Test Engine

The granularity of user control within this application was more refined than a commercial application to support the detailed test activities performed. Amalgamation and automation of several command sequences was implemented in the commercial software to remove the responsibility of P2P overlay connection from the user. This supports a mobile application 'good practice' of making minimal demands upon the user. Access to application functionality was primarily via the normal soft key 'options menu' as directed by the standard Nokia S60 user interface. The menu presents a set of options to replicate the serialised sequence of stages necessary for a node to join a P2P network. Stages include establishing a link with 'SysProxy' and attempting to contact a known node from the routing table as previously described in section 4.4.1. Feedback to the user on node status and activity is presented either via a screen message or recorded into a handset log file. Both feedback methods offer information in different situations dependent upon the a 'real time requirement' and the quantity of data to be analysed. A log file is the most detailed recording method, but only available for interrogation once the application has been closed. In contrast, the screen gives more immediate feedback but timing control, persistence and quantity of data shown are subject to operating variations.

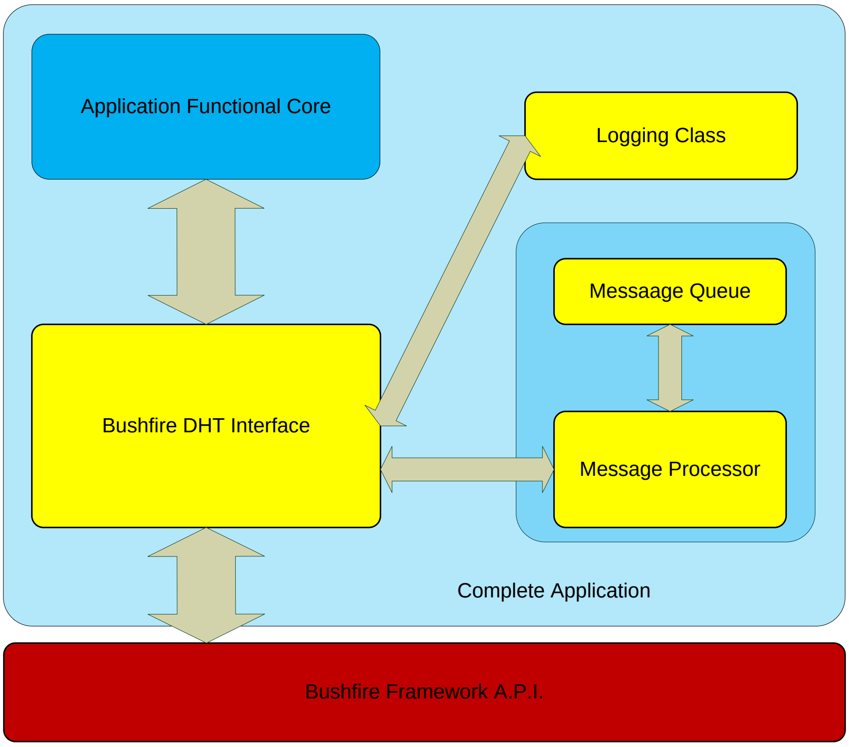


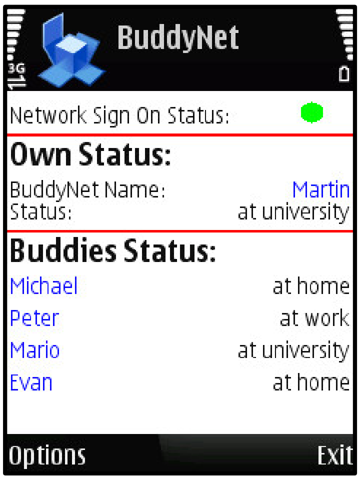
Figure 5.3: Bushfire Test Engine Functional Block Diagram

Figure 5.3 illustrates a functional block diagram of the application. A typical application will consist of code associated with the Nokia APIs, the Bushfire framework and the application itself. Those classes associated with the Nokia API have been omitted from all functional diagrams in this chapter for clarity. Functions associated with integrating with the Bushfire API are contained within the class 'Bushfire DHT Interface'. The use of these functions is as described within section 4.4.3. All data communication between the Bushfire framework and the application core functional class was via this DHT interface class. Additionally, integrated with this class are functions relating to application logging (for debugging purposes) and message buffering (to ensure no loss of data). Buffering of all incoming message data is necessary (via Message Processor and Queue) to overcome processing limitations and variations in the connectivity and speed of network interaction. Given the minimal complexity of the Test Engine application itself, code relating to the core functionality was integrated into a single class.

The Bushfire Test Engine application became an integral aspect of the 'iterative' design process employed throughout later developed application projects. To determine if newly implemented functionality was working correctly and that it met the user's expectations, the application became the initial host of the appropriate code acting as a 'Sand Box' test platform. Assuming functional and user tests were completed successfully, the code was transferred across to other, more complex, applications. Hosting code on the less sophisticated Test Engine application in the first instance helped to reduce complexity and simplify debugging activities.

5.2 Handset Application: BuddyNet

Best described as a 'proof of concept' application, 'BuddyNet' [1] illustrates the mobile P2P networking technology clearly using well established concepts. BuddyNet is a stand-alone application designed to mimic the 'status notification' concept prevalent within many Instant Messaging (IM) networks. Figure 5.4 illustrates the main status screen of this application.



BuddyNet

Network Sign On Status: ●

Own Status:

BuddyNet Name:	Martin
Status:	at university

Buddies Status:

Michael	at home
Peter	at work
Mario	at university
Evan	at home

Figure 5.4: Screen Image of BuddyNet Application

Focusing on people already known to you, 'Buddies' are added via the handset contacts directory and status information chosen from either a predefined list of options or a freeform text entry. Users simply initiate a refresh command to download (via the Bushfire framework) all the latest status information from their known contacts. The majority of information is conveyed to the user by text with some visual status indicators. BuddyNet is intended to run as a background application, viewed when the user requires. This usage scenario plays to the strengths of a P2P node behaviour and the Symbian operating system where multiple applications can run in parallel. Although primarily demonstrating the P2P framework, the single-screen layout design ensures clarity of comprehension for the user. As intended, users are not aware of configuration settings associated with the P2P network.

Figure 5.5 illustrates the functional design of the application presenting a more complex application than the Framework Test Engine. However, BuddyNet replicates the DHT interface infrastructure from this application; the API design was identical to that described in sections 4.4.3 and 5.1. These functions (Bushfire DHT Interface, Logging, Message Processor / Queue) form the outer interface to the core application. Illustrated by five functional segments, the core application section focuses on provision of the services and features that form the application as seen by the user. BuddyEngine represents the main class of this section, responsible for the cohesion and co-ordination of 'helper' classes tasked to implement specific functions. These classes either implement application-specific tasks or the method calls associated with accessing standard handset functionality (such as sending an SMS message or interrogating the contacts directory - refer to section 4.4.5). BuddyRegister represents classes responsible for organisation and storage of the data downloaded from other nodes on the P2P network. The data

[1] BuddyNet is not associated with TATA DOCOMO BuddyNet Service operated in India, launched 2010 [64].

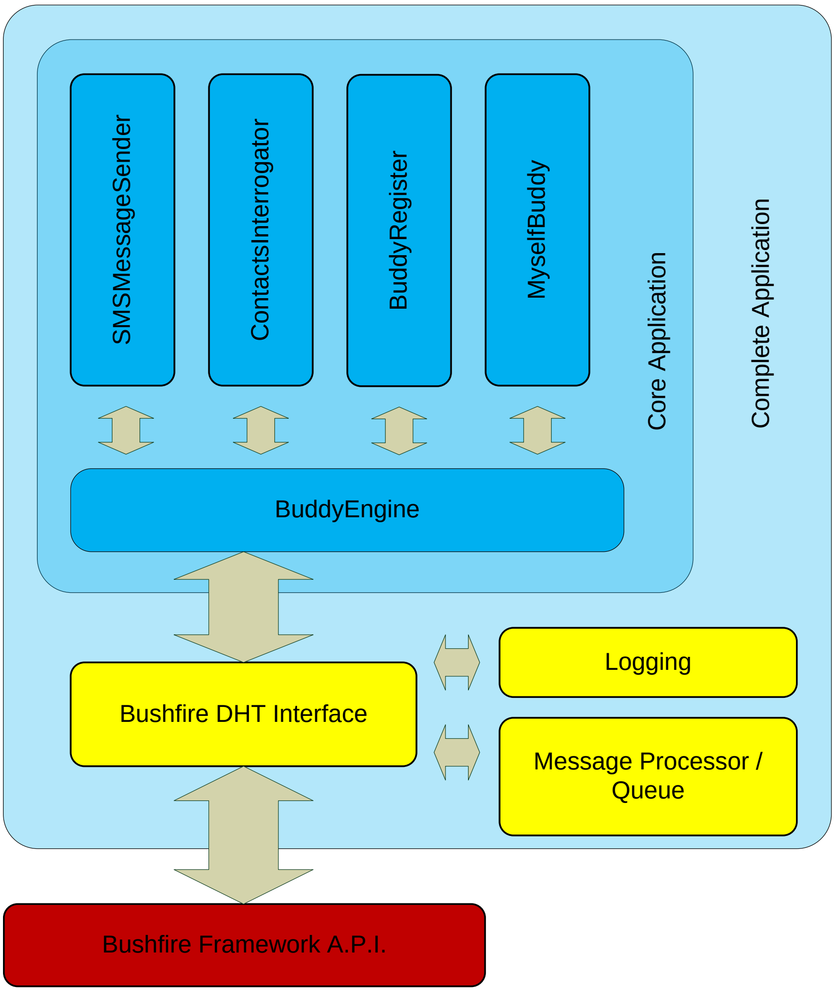


Figure 5.5: Functional Block Image of BuddyNet Application

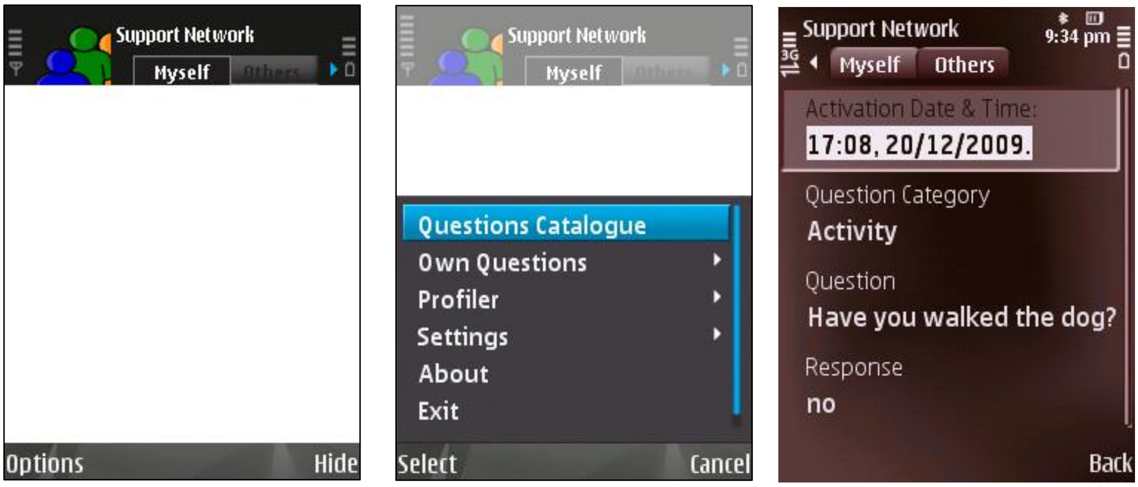
stored within the network is simply a key:value pair set as telephone number:status respectively. This is in line with the requirements described in section 4.1. The MyselfBuddy class is a logical extension of the Buddy and BuddyRegister classes. This stores personal telephone number / status information for transmission to the other nodes on the network (as a key:value pair).

BuddyNet has a plausible use case scenario in both a business and social context. The application was demonstrated at a conference as a supportive illustration of using Bushfire and its API [7]. Feedback from users in these domains was obtained and incorporated into the next commercial grade application.

5.3 Handset Application: SupportNet

SupportNet is an application designed to aid the natural support network of family and friends. SupportNet was developed with the intention of being the first commercial grade application to use the Bushfire framework. SupportNet extends the concept of BuddyNet beyond a status monitor to an interactive tool by communicating and coordinating the activities of carers/supporters. Inspired by a family caring for an elderly relative, SupportNet is part of a growing trend of applications that are aiding the provision of care and support to people in the community [150, 77]. Whilst not a traditional medical application, such as in the form of telemedicine, it could potentially contribute in other ways. For example, it promotes better well-being by removing the feeling of isolation or assisting in the compilation of time-dependent data.

Conceptually, supporting family members (the ‘Supporters’) will be part of a group associated with a focus person (e.g. the elderly relative). The application administers both groups of people through a tabbed interface with a different screen for each. The user conceptually adopts the role of ‘supporter’ or ‘the supported’ depending upon which tab screen they are viewing. Therefore the application is a conduit for both roles to record information about themselves and monitor that of others. The following paragraphs and figures illustrate the dual-purpose nature of the application.



Activation Date & Time:
17:08, 20/12/2009.

Question Category
Activity

Question
Have you walked the dog?

Response
no

Figure 5.6: Screen Images of SupportNet: ‘Supported (or Focus) User’ Screens

The ‘supported person’ role has a simple interface with most interaction performed either via menu options or dialogue boxes (see Figure 5.6). This role requires the handset owner to answer questions presented via dialogue boxes on their device at certain times. Questions are pre-configured on the handset prior to any monitoring phase. Questions can be specific and related to their presentation time (e.g. Have you eaten your breakfast?) or random both in question nature and time (e.g. What have you read in the newspaper today?). More crucially, answers to these questions are stored in a local handset file and replicated on the Bushfire P2P network. Supporters of the focus person can then formulate a profile of their day and determine if a telephone call or visit is required.

The ‘supporters’ view of the application is similar in design to the BuddyNet main screen (see Figure 5.7). This view comprises persons added from the device contacts list whom the ‘focus user’ (i.e. the device owner) would like in their support group. Selecting a contact and drilling down reveals more information about them via the given answers to the presented questions. A supporter is also able to ‘inject’ a question into another person’s schedule. This ensures questions are tailored to specific events in the focus user’s routine. Administrative functionality associated with the supporters is also presented through this tab screen.

Extensibility was promoted and maintained by the use of an XML formatted text file as the source of the questions on the focus person’s handset. The proprietary, application specific XML format records question properties such as question text, format (e.g. yes/no, free text) and answer. It was noted commercially that, as well as getting the user community to extend the application by broadening the question base for this particular use case, other question sets could be developed for different scenarios. For example where team support is a strongly influencing factor in success rates, the application could promote peer cooperation through motivation or competition (e.g. a weight loss support group).

Figure 5.8 illustrates the functional design of the SupportNet application. Similarities are immediately noticeable, with the functional block diagram describing the design of the BuddyNet application (see Figure 5.5). This design re-use was intentional to permit a faster development time as some similarity in requirements existed.

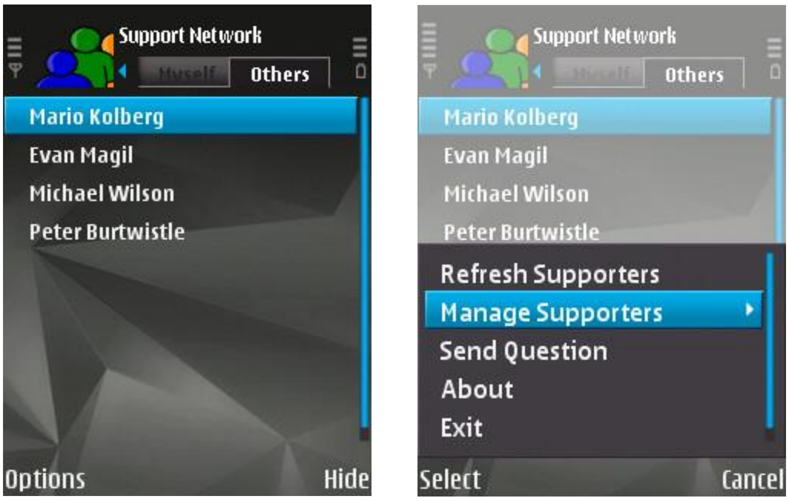


Figure 5.7: Screen Images of SupportNet: 'Supporters' Screens

This also reduced problems associated with debugging a new application. The now well-defined interface structure between the Bushfire Framework API, DHT Interface and SupportNetEngine class replicates functionality introduced within the BuddyNet application. The SupportNetEngine class again performs a coordinating role for the core application classes, both with previously integrated code (e.g. SMSMessageSender) and newly developed tasks such as 'Scheduler' or 'TaskManager'. The Scheduler class incorporates a logic engine to calculate when questions could be presented based upon sets of rules defined by the user. Promoting the concept of flexibility, the classes 'TaskManager' and 'TaskBank' deal with objects that in SupportNet are 'Questions', but in other applications could be any data object type. Referencing extensibility discussed previously, the 'Taskbank' class accepts question data loaded during application start-up from an XML file on the handset. Other configuration data could also be stored using this method and loaded into the appropriate settings class to permit modification of the application operation.

5.4 Security Design Considerations

Security design considerations for mobile applications work in unison with concepts introduced in section 4.4.7 relating to the Bushfire framework. As also highlighted within section 4.4.7, security considerations remained a considered topic rather than implemented functionality due to the limited time constraints expressed within chapters 3 and 4. Application security can be discussed with respect to both the link between application and framework and also the link between different instances of the same application. Foremost, the framework API should be designed to ensure that any interaction or interface between the framework and the application happens in a restricted and controlled manner. With the same application and framework host software installed on every peer handset, any exploited weakness has the potential to propagate widely and easily. Depending upon the nature of the attack, the security breach can remain within the application layer, using the overlay purely as a propagation medium. Alternatively, the framework could become the target in order to prevent node communication. Therefore both the framework developer and application developer must assume equal responsibility in solving security related issues.

With reference to the paper by Kirkman et al. [21], a summary of the principal, relevant security concerns appropriate to the application level is given below:

- 'Identity theft' or 'Sybil attack': A malicious application on a node forges its identity in order to easily access restricted data by 'user masquerading'. This could be implemented by overriding the node authentication mechanism or data packet spoofing on the overlay.

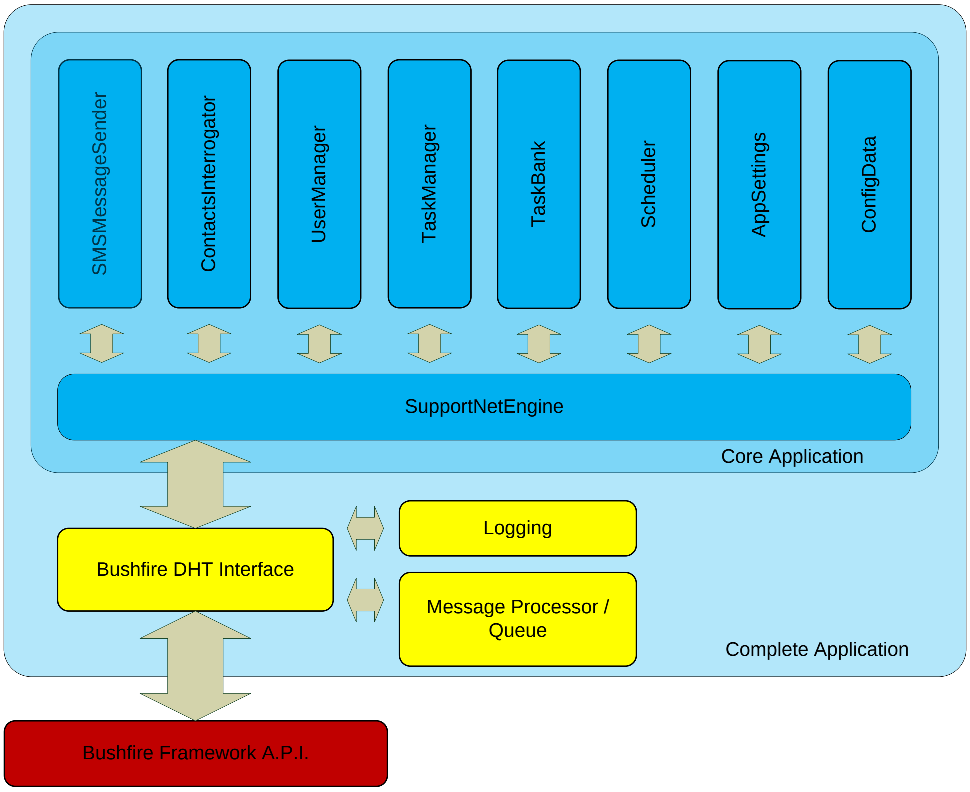


Figure 5.8: Functional Block Image of SupportNet Application

- ‘Unauthorised data accessibility’: As user data is dispersed across the multiple nodes participating in the P2P overlay, adequate design diligence is required to ensure individual users cannot read, direct from their handset, private information belonging to other people. Encryption of user data to an adequate strength with an individual unique key will resolve this security weakness.

- ‘DDOS attack’: DDOS attacks are unlikely to target the application themselves. A more likely scenario is that a compromised application installed on many nodes will target the Bushfire framework itself, targeting known weaknesses such as the single SysProxy server application. This attack could be rendered ineffective if the ‘multiple SysProxy’ solution as discussed in section 4.4.6 is implemented.

- ‘File poisoning’: Attacks on an application network’s content would be minimised by reducing the opportunities for unauthorised peers to place corrupt data on legitimate nodes. Solutions would include closer scrutiny of nodes that can upload data or some form of data validation to ensure it is legitimate.

- ‘Eclipse’: This attack would require mass collusion of infected nodes to corrupt a previously clean node. This approach could be used by malicious code networks to ‘jump’ applications that use the same underlying overlay transport mechanism.

Section 7.4 raises application security as a topic necessary for future consideration when application development for the extensive framework test is commissioned.

5.5 Summary

The three applications presented within this chapter illustrate the iterative development process that has occurred to produce more stable, functionally diverse software. SupportNet alludes to further development possibilities where any digital data capable of storage within a standard Symbian C++ buffer would be appropriate for storage within the P2P framework network (assuming the integration of functionality for full data segmentation and realignment is completed).

Chapter 6

Optimisation of P2P Framework Performance

Optimisation of a working system is the next logical step in the evolution of its design. P2P networks as well as mobile technologies are no exception to this process. Both domains have been the subject of research effort investigating techniques surrounding theoretical operating algorithms [109, 171], practical software implementations [84] and hardware configurations [41]. The Bushfire Framework has also received research effort to investigate possible routes to improved performance. Authors such as Li et al. [130], in addition to other evidence presented in section 3.1, have cited many challenges facing mobile P2P network design and operation. Some of these include variable telephone network connectivity and performance. However it could be argued two themes have more impact than others: energy consumption and bandwidth utilisation. Both of these factors are under the direct control of designers as opposed to reacting to minimise impact as in the case of phone network disconnections. They can influence at a fundamental level how successful a P2P network is as they directly affect how long a node can participate and its perceived cost to operate. In reality, energy consumption and bandwidth are closely related. A simplistic view is that the more data that has to be sent across a network connection equates to a longer period of time when the high energy consuming transmitter in the handset is active. Li et al. [130] even assert that device battery life and limited bandwidth hinder typical P2P overlay operations to the point that it is infeasible working in a mobile domain. However, this chapter discusses the optimisations particularly investigated for the Bushfire Framework that could also be applied more widely.

6.1 The Energy Shortage Explained

Introduced within section 3.1.1, energy consumption relating to the operation of a mobile P2P network is a parameter that has received some level of research interest. The focus of these activities has primarily concerned the behaviour of the software on the handset. However, in reality two other factors have a major influence on energy consumption: the handset hardware and the network configuration. The three aspects that complete the whole picture relating to energy consumption are presented in the subsequent sections; the reality, as will be shown however, is that only the software behaviour is under the control of the developer.

6.1.1 Energy Consumption from the Handset Perspective

Energy consumption within the handset hardware is an obvious investigation subject, yet despite advances in technology and operating practices the traditional mobile energy store, the battery, remains one of the limiting operating factors [61]. Although technologies introduced into the marketplace over a decade have improved battery storage capacity by an estimated 80% [96], this has been overshadowed by the increased demands placed on the battery to run more complex hardware. Whilst unbiased power consumption statistics are difficult to obtain due to manufacturer secrecy, multiple test variables and unlike comparisons, it is possible to obtain general trends through academic research. It has been shown that power consumption has approximately doubled in the transition from earlier generation devices to the latest '3G handsets' [96]. The previously typical 1 to 2 watt power consumption [96] of the earlier devices has not doubled solely because of more complex 3G telephone technology, but has been augmented through the inclusion of other integral devices (e.g. colour/touch screen, bluetooth, GPS, etc.). This is supported by the findings of Carroll et al. [40] who state that screen and GPS technology have been the highest energy consumers on their test handsets after the components used to make a GSM telephone call. Other simple measurements taken using the Nokia Energy Profiler Software application [11] illustrate how influential on power consumption the screen operation is. For a Nokia N95 8GB handset (released 10/2007) there is a 250mW difference in power consumption between no illumination and maximum brightness. Carrol et al. conclude that the 'most effective power management approach on mobile devices is to shut down unused components...' [40]. For example from their measurements, turning off just the BlueTooth transceiver could save a minimum of 36mW on a Google Nexus G1 handset (released 01/2010). And finally, to complicate matters further, the individual usage profile of the device, its location and even typical operating temperature will affect the battery performance [44].

Users are becoming more aware through manufacturer education that there is a trade-off to be made between functionality and battery life [44]. Yet where the perceived value of the functionality is high (such as data intensive applications) this appears to take precedent, supporting a view that 'up to a threshold' users will accommodate the failings of battery technology [22]. Therefore whilst ultimately this energy supply is still limited, extended operation times can be supported through more ingenious operation of software that operates on the device.

6.1.2 Energy Consumption from the Mobile Network Perspective

Operation of more complex integral devices is not the only contributor to battery drain of mobile telephones; part of the problem is inherent to the design of handset participation within the mobile network itself. The radio transmitter associated with the handset consumes the most power out of all components within a handset. It is stated by Haverinen et al. [25] that theoretically (and under traditional operating regimes) 3G network technology is efficient at preserving handset power consumption. Achieved through initialisation of a handset 'idling process' when there is no network traffic, smaller amounts of energy are consumed because components normally involved with the radio transmission are not used. In the case of traditional handset software (client/server request format), constant demands are not made of the network and therefore the device can enter the lower power idle state when not servicing requests. Assuming typical behaviour patterns of a user over the period of a day, there will be a significant period of time when the handset is in idle mode and therefore operational life is extended through energy preservation. However in contrast, Haverinen et al. [25] state that any example of a 'push data' application (e.g. push email, VOIP) operated in conjunction with a firewall or NAT device will increase device power consumption over an extended period. This is because these applications require an 'always reachable' state to be maintained both with the connections to the NAT and mobile network maintained. In short, it is equivalent to maintaining the handset in the constant state of a telephone call and thus usable battery life times reduce from the typically quoted lengthy 'standby' figure to the much smaller 'talk time' figure [81]. Unfortunately, because of the intrinsic behaviour of P2P nodes, their profile matches the 'push data' application category.

Another behavioral aspect of data transfer on the 3G data network that has a negative impact on a handset's energy efficiency is referred to by Balasubramanian et al. as 'Ramp Energy' and 'Tail Energy' [131]. These terms refer to the energy necessary to establish or close a communication session but do not actually transfer data. (i.e. energy for the 'wrapper' network data that encapsulates the user data.) Balasubramanian et al. [131] indicate that up to 60% of energy in a 3G network data transfer is consumed in these stages of the connection. With typical P2P network activity involving repetitive network calls to a node (e.g. routing requests, routing table refresh) the implication is that the majority of energy used is not for the actual transfer of the user data. Therefore the focus of development activity should be on optimising the user data transfer.

Furthermore, such has been the impact of handset operating behaviour on the network that two reports have been produced by the Signals Research Group (Canada) in consultation with many leading Network Operators and Equipment Manufacturers to document and analyze the situation [36, 78]. The reports concentrate on the issues relating to network activity and data consumption, but as a consequence also reveal the impact upon energy consumption in the handset. During the rise in adoption of smartphones, these reports have shown that although data consumption on the mobile networks has increased, the signalling (control) traffic associated with delivering this data is increasing by 30% - 50% more [35]. In relative terms alone, the implication is that disproportionately more energy is being consumed now by the handset performing data-related activities. Therefore with this operating environment for mobile handset applications, any improvement in efficiency resulting from framework optimisation will have a positive impact.

6.2 Existing Ideas for Energy Optimisation

To consider ideas for implementation within the Bushfire Framework a review strategy was followed initially: evaluating hardware, software and operating techniques. These three subject areas have been the past focus of optimisation processes and demonstrate a range of approaches to consider. The following sections will elaborate both the nature and applicability of current solutions as well as appropriateness for Bushfire's requirements.

6.2.1 Energy Optimisation with Hardware

When considering energy optimisation, reducing hardware power consumption is the first obvious approach. Unfortunately some 'high power' elements of a handset are out of scope and control of an application developer, either because they are user-selected (e.g. GPS, screen brightness) or necessary for operation (e.g. 3G radio circuits). For example, the power consumption difference between handset standby and GPS active has been measured as 597mW on a Nokia N95 handset [41]. Obviously this equates to a significantly shorter P2P network participation times when GPS hardware is operational. However an approach could be to direct the user (automatically or voluntarily) to operate the handset in a certain manner to prolong the operating time period. For example 'auto-switching' to a WiFi connection when in range of a WLAN or detecting when at a location that may have a mains power source. In a study by Nurminen et al. [81] it was demonstrated that WiFi is three times more energy efficient compared to 3G for file transfer so there can be tangible benefits to these approaches.

6.2.2 Energy Optimisation with Software

Limited by system constraints, the alternative approach to hardware optimisation propositions are typically clever and concern the construction of the software. Saxe [110] concisely describes software's role in a system's efficiency as undeniable: 'the salient part is really the way in which software interacts with power-consuming system resources'. Techniques for 'power efficient software coding' are discussed in many sources such as manufacturer developer guides [13], web casts [30, 5], articles [110] and forums [127]. For a particular scenario a combination of the generic techniques and detailed evaluation will yield results as the previously discussed papers have implied. Key examples from both categories are shown in the following list:

i Program using code efficient algorithms that minimise demands upon the processor. For example do not use the 'polling' construct as it will keep processor(s) active; instead utilise system events that 'wake' the processor when necessary.

ii Gather network data and transmit it in bursts to avoid excessive use of the network interface. Utilising a batch processing approach can also be applied to processor usage, file system read / write activities, etc.

iii Allocate an energy budget to an application and a quota to individual tasks within a software application. For example search queries could be routed based upon energy levels remaining on nodes within the network. Also adopt tasks to favour handsets having their battery charged for energy intensive activities such as a routing table refresh [67].

iv Use data compression to reduce the amount of data transmitted across the network. However as more energy could be used in the compression task, an evaluation of exact circumstances would be necessary. [67]

v Make changes to the P2P protocol to make it more 'energy consumption aware'. For example a node could partially 'sleep' after a specified time period and fully 'wake up' only when communication directly intended for it is received. Significant power savings could be possible if the protocol implements batch sending to allow data transfer periodically to support sleep capability [170].

vi Perform load balancing of nodes on the network by 'selectively dropping incoming requests' on an individual node dependent upon its stored battery energy reserve. [83]

vii A node participating in P2P network activities should only serve other nodes' requests when it is performing a download operation itself. The justification for this is the minimal increase in power consumption by the radio circuits between upload and download of data simultaneously. [81]

The techniques presented in this section whilst offering the desired energy savings also frequently have side effects that need to be compensated for. For example an impact on the project development phase is possible due to extended design stages necessary to complete energy efficient software code. Additionally item iv. may use more energy compressing the data than is saved in the reduced transmission time. The side effects seen with the specialised solutions are more likely to be complex to understand and closely coupled with the algorithm or technology they are related to. For example, item vi. introduces a problem described by the authors as virtual churn where nodes are actually 'online' on the P2P network but others see them as 'offline' because of a non-reply to their request.

6.2.3 Energy Optimisation through Bandwidth Optimisation

Techniques in energy usage optimisation have also involved investigation into bandwidth consumption. Reduction in data sent across the network will reduce power consumption through less use of the device transceiver. Data compression (using ZIP or RAR format for example) is one approach, but this is frequently applied to the user data stored within the P2P network and therefore if re-applied will have limited benefits. Considering at a more fundamental level the P2P overlay algorithms employed, the number of nodes participating in the overlay network and the way in which they are configured can all affect bandwidth consumption. For example the frequency of routing table maintenance will directly impact on the quantity of data transferred. However another 'trade-off' is necessary at the expense of accuracy and latency; the more accurately node joins / leaves are to be recorded, the more communication must take place between nodes. Latency is consequently affected because, with inaccurate routing tables, it takes a node longer to respond to a data request. Both accuracy and latency are very pertinent considerations for a network with mobile handset nodes. This is principally because Churn is a significant factor and latency values are already on average higher than traditional network connections.

The earlier P2P network designs relying on a 'broadcast' approach had the weakness that bandwidth consumption increased significantly as the number of nodes scaled up. Additionally there was an impact on the participating nodes in terms of increased computing resource (energy) required to process these requests [120]. However later more advanced overlays, such as Kademlia used in the Bushfire framework, do intrinsically accommodate larger networks more efficiently. Furthermore, the selection of a multi-hop overlay implementation rather than a single-hop has an almost constant bandwidth impact as the number of participating nodes in the overlay increases. Bandwidth consumption results for a standard data lookup, retrieval and maintenance activity provided from tests by Buford [6] confirm this statement: a single-hop overlay network in the order of 1 million nodes consumes nearly 12 times the amount of data of an equivalent multi-hop overlay network. Given that certain optimisations have been included inherently already, Bushfire will have to select the best combination of these parameters to give an optimised mobile network performance.

6.3 Energy Usage Optimisation Applicable to the Bushfire Framework

Whilst obviously dependent upon an exact usage profile, an acceptable target scenario of a 'once per day' charge cycle for a handset is assumed for operating a data intensive application. Currently, as a generalisation, handsets running 'push data' applications will achieve less than this. Ultimately the operating time also depends upon the handset battery capacity (i.e. stored energy) and location with respect to phone network cell antenna (i.e. handset transmitter power required). These are parameters that are usually not part of a user's conscious decision process when running 'push data' applications on their handset, yet the desire to use this class of software remains strong. Therefore the requirement for the Bushfire Framework could be summarised as 'increase device operation time on the P2P overlay network compared with the initial reference (BUTE) implementation through smarter software usage'. Any improvement targeting the framework will yield positive results irrespective of handset design. It is assumed that the handset would behave as a typical node in a real P2P network by actively participating in both data storage and message routing activities. The following sections discuss the investigations conducted specifically on the Bushfire Framework.

6.3.1 Energy Optimisation Experimentation

Knowing that energy is consumed within a handset when its transceiver circuits are active, the data that is transmitted to or from a handset (node) when participating in a P2P network must therefore be responsible for energy consumption. As has been discussed in previous chapters, many operations within a P2P network are responsible for data flow such as user uploads, search queries, node joins or routing table refreshes. The first two operations suggested are pseudo-random in a typical network, dependent totally upon the popularity and activity levels of its participating users. The latter operation however is not user-influenced, being intrinsic to the operation of any P2P network. This refresh parameter can therefore be more easily controlled to modify data flow. This parameter is the focus of experimentation discussed within section 6.3.2. The 'node join' function is conceptually between the two of these previous situations: the user determines when the request happens, but controlled by the node logic as to when the node actually joins the network. A 'node join' can be both an initial registration in the network or a 'reappearance' after a time of non-participation. Currently, typically when a node joins an overlay network, the routing table is refreshed immediately. Whilst within the traditional desktop PC environment this subtle timing / sequencing variation may make no discernible difference, for the mobile domain the impact on energy preservation may be noticeable. This routing table refresh sequencing is the focus of experimentation discussed within section 6.3.3.

General Methodology of Experimentation

Both experiments discussed in sections 6.3.2 and 6.3.3 were conducted using a similar configuration: a mix of hardware and software to replicate a very small scale P2P network. The observations were conducted using manual techniques, configuring the network of nodes and monitoring software as appropriate for each parameter variation. An outline illustration of the experimental structure is shown in Figure 6.1.

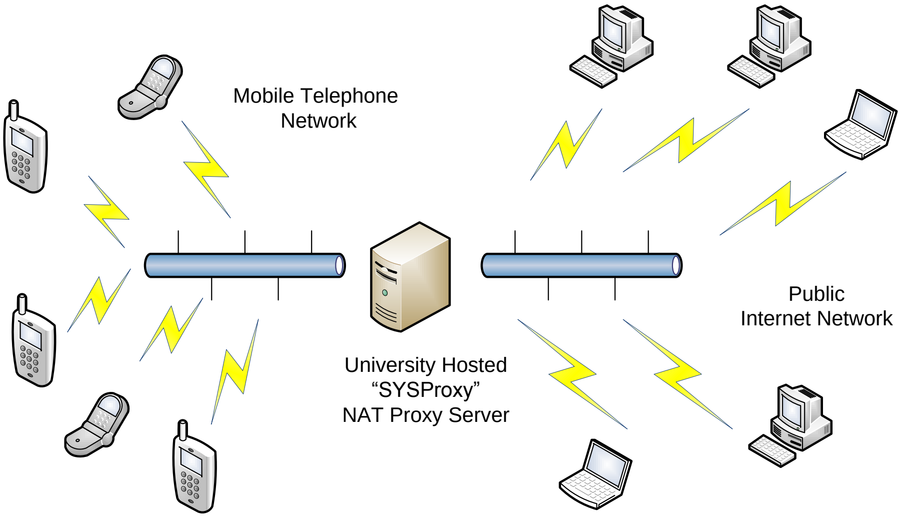


Figure 6.1: Outline Illustration of Experimentation Configuration

Further details are offered in a description of the composing principal elements as follows:

i Nokia S60 series (3rd edition) mobile telephones. There were 5 telephones, each a different model, and configured on a variety of operator networks.

ii Nokia/Symbian handset software emulator running within a Windows XP virtual machine environment.There were 5 virtual 'emulator' telephones operating on 3 physical host Windows PC machines (Windows XP & 7).

iii 1 Windows XP PC machine configured with a public internet IP address. This machine hosted the 'SysProxy' software application and was located within the University of Stirling Computing Science project laboratory

.

iv Nokia Energy Profiler Handset software application [11]. A proprietary application produced by Nokia for recording numerous key parameters of a telephone as a background task during operation.

v SupportNet - Bushfire Framework P2P application. An application developed specifically to use the P2P framework under consideration (see section 5.3).

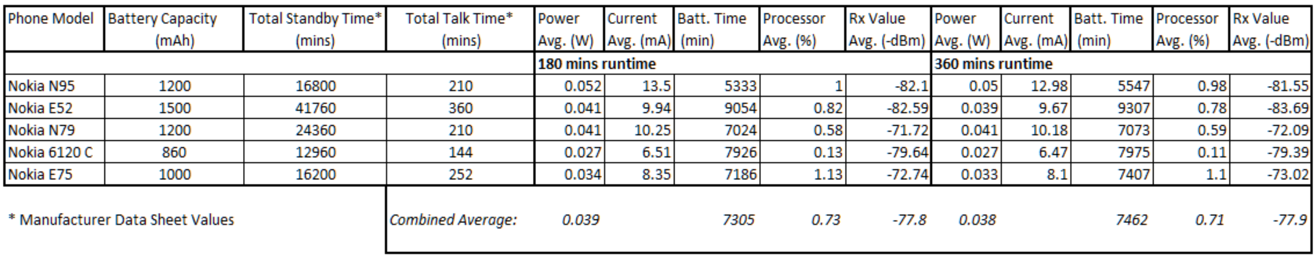
The 5 telephones used in the experiments, whilst all Nokia S60 Series devices, represented a selection from the manufactured range available. There were a variety of form factors (including screen size), battery capacities and processor capabilities. All devices, however, were capable of running the P2P and Energy Profiler application and equipped with 3G network access capabilities. These handsets were the focus of the energy monitoring observations but, in order to increase the number of participating nodes within the network, handset emulator software was operated from virtual machine hosts. The emulator software behaved identically to the handset configuration once the application had been loaded and therefore offered an inexpensive method to increase node numbers. For obvious reasons the two hardware solutions accessed the internet by different means but both worked successfully with the publicly network addressable 'SysProxy' application.

When conducting the experiments discussed in the following sections, two software applications were used: SupportNet (as discussed in section 5.3) and Nokia's Energy Profiler [11]. SupportNet was selected as it represented a pragmatic use case application using the Bushfire framework, suitable for obtaining realistic data on performance. The Nokia Energy Profiler application was used to monitor parameters most applicable to the experimentation objectives: power consumption (watts), current consumption (Amps) and Processor activity (percentage). Ultimately average power consumption gave the clearest indicator of operating performance. However current consumption was used with knowledge of the handset battery capacity to give an approximate 'battery lifetime' calculation - an important, easily interpretable usage metric. Processor activity offered an opportunity to see the relative impact of running the framework on the hardware on handsets with dedicated mobile devices.

The following table (6.1) illustrates the data that was retrieved from telephones being used within the experiments. This data represents the control data and has been compared against the public technical specification figures claimed for the devices. Differences are clearly inferable between the manufacturer and observed data, illustrative of the diversity of variations possible in configuring and operating the devices. Whilst data was not available on how the manufacturer figures were measured / inferred, the control figures were taken primarily in standby mode, Bluetooth on and Nokia Energy Profiler application running (to record the measured data). All measurements were taken from an identical urban environment with strong network received signal strength (-78 dBm average across all devices compared with figures of -102dBm to -87dBm for the GSM standard reference receive sensitivity [86]). The Bluetooth radio device was left operational during the observations to represent an element of 'realism' with the likely use case scenario. Whilst Bluetooth remained a constant influence on results, the operation of the screen was not as controlled. As previously stated, from simple observations on a Nokia N95 handset (which had the largest screen size in the test hardware selection) there was as much as 250mW alteration in power consumption between the screen on and off. Notwithstanding this circumstance, attempts at achieving a consistent operation across all devices were made through similar 'screen saver' configurations as well as reducing human handset interaction to a minimum. The impact of initial screen illumination (when starting the experiment in section 6.3.2) was also minimised further through extended run times. Test data indicated that a running time of 3 hours gave a good compromise between the heavily influenced data recorded during the first 30 minutes and the almost unaffected data set when averaged across a 6 hour observation. Finally, averaging of recorded data across the 5 handsets further minimised the impact of any one particular handset hardware configuration (such as screen size) on the results. For example, the Nokia N95 handset screen size was 108% larger than the Nokia 6120 Classic handset and yet the battery capacity was only 40% larger implying a smaller operational time for the Nokia N95 before any other considerations. Further discussion relating to perceived limitations of the experimentation is presented in the following section, 6.3.1.

Limitations of Experimentation

Experimentation associated with evaluating P2P networks in general does have difficulties and limitations [20, 99]. The work conducted for this chapter was no exception to this observation. A number of reasons to be considered for this work are offered in the following list:



Phone Model	Battery Capacity (mAh)	Total Standby Time* (mins)	Total Talk Time* (mins)	Power Avg. (W)	Current Avg. (mA)	Batt. Time (min)	Processor Avg. (%)	Rx Value Avg. (-dBm)	Power Avg. (W)	Current Avg. (mA)	Batt. Time (min)	Processor Avg. (%)	Rx Value Avg. (-dBm)
				180 mins runtime					360 mins runtime				
Nokia N95	1200	16800	210	0.052	13.5	5333	1	-82.1	0.05	12.98	5547	0.98	-81.55
Nokia E52	1500	41760	360	0.041	9.94	9054	0.82	-82.59	0.039	9.67	9307	0.78	-83.69
Nokia N79	1200	24360	210	0.041	10.25	7024	0.58	-71.72	0.041	10.18	7073	0.59	-72.09
Nokia 6120 C	860	12960	144	0.027	6.51	7926	0.13	-79.64	0.027	6.47	7975	0.11	-79.39
Nokia E75	1000	16200	252	0.034	8.35	7186	1.13	-72.74	0.033	8.1	7407	1.1	-73.02
			Combined Average:	*0.039*		*7305*	*0.73*	*-77.8*	*0.038*		*7462*	*0.71*	*-77.9*

* Manufacturer Data Sheet Values

Table 6.1: Control Data Summary Results: Telephone Handset [152, 153, 154, 155, 156] Manufacturer Data for Standby Times and Measured Battery Life Data (+ other control data).

- Low number of participating nodes within the test P2P network. Given the description from chapter 1 indicating that node numbers in the thousands are possible with a theoretical limit of 14535 set by the design of SysProxy (see section 4.4.6), the test scenario is very limited. For a practical test, the limitation of numbers of handsets is obvious but not critical to the success of the initial observations.
- Independent validation of observed data recorded by the Nokia Energy Profiler application. Research has shown there are some similar energy monitoring applications available (e.g. PowerTOP [157]) but not on the Nokia hardware platform under investigation. As the application is proprietary there has been no independent validation of its accuracy although it has been quoted as used in several published academic and industry articles [131, 36].
- Difficulty in measuring energy associated solely with P2P network activity. As has been fully discussed within earlier sections (6.1, 6.1.1, etc.) other components within a mobile device are responsible for energy consumption whether controllable or not. An example such as the handset screen activation was difficult to avoid as it was necessary for operating the software under test. As a counter-argument to this, it could be argued that screen operation is a valid contributor to the results given it is required. Another example is the power consumed by the telephone radio transceiver within a handset. This is subject to variation primarily because it is using GSM cellular telephone technology. Working in unison with the cell base station, the transmitted power of a handset has been designed to vary according to received signal strength, which is most notably affected by location and atmospheric conditions.
- Inherent difficulty in conducting an experiment where there are so many variables to account for. Any attempt to fix variables with constant data in order to focus on the property of consideration can result in unrealistic or problematic operating scenarios. This in turn yields results data with a limited scope of interpretation.

Consideration has been given to addressing these limitations within the tests that were conducted. For example to address the issue of the low number of participating nodes, aside from considering acquiring more handset devices (with the obvious complexity problems), the use of a P2P network simulator such as P2P Sim (using C++ [76, 99]) was considered although its current status appeared inactive. However, simulation had been initially dismissed from this level of experimentation due to implementation constraints (language compatibility - Symbian C++) and limited scope (primarily protocol testing - which is not necessarily handset framework testing). Nonetheless, the experimentation that was conducted was selected to allow for these limitations where possible and maximise the 'return on investment' of evaluation effort. For example both experiments described in sections 6.3.2 and 6.3.3 investigate influential parameters that affect data volumes and flow within the P2P network. Consequently there is an impact on the device energy consumption, but importantly, one that is independent of user data. Conducting tests independent of user data is advantageous as a record of comparable results can be more easily obtained. When injecting user data into a P2P network it was recognised that achieving repeatability as part of a test scenario is a difficult task.

In reference to the selection of Nokia Series 60 Handsets for the experimentation, they did have an advantage that the Energy Profiler application was able to operate as a background task during normal phone operation, thereby allowing measurement recording to occur during other activities. However, an unavoidable consequence of this was the implicit inclusion of the Energy Profiler application itself within the results obtained. This effect was diminished because of its applicability across all observed results and when analysing data trends rather than absolute values. Further generalisation of the data was performed by averaging values across the 5 devices under test, thereby equalizing the influences of individual, differing handset performance. These latter remarks emphasize that, given the limitations detailed in the aforementioned list, the observations presented within sections 6.3.2 and 6.3.3 remain valid when interpreted as trend data rather than absolute figures.

6.3.2 Experiment 1: Energy Consumption Affected by Frequency of Routing Table Refreshing

As introduced in section 6.3.1, the focus of this experiment was to observe the impact of the node routing table refresh frequency upon the energy consumption of a mobile telephone handset. In short, for each routing table refresh (which heavily dictates routing accuracy), energy is consumed. The motivation for conducting the test was to discover the details of the relationship between handset energy consumption and routing table maintenance. Whilst it is also recognised that churn of participating handset nodes can contribute to routing table inaccuracy, the impact of this aspect was not assessed in this experiment.

Experiment 1: Configuration and Results

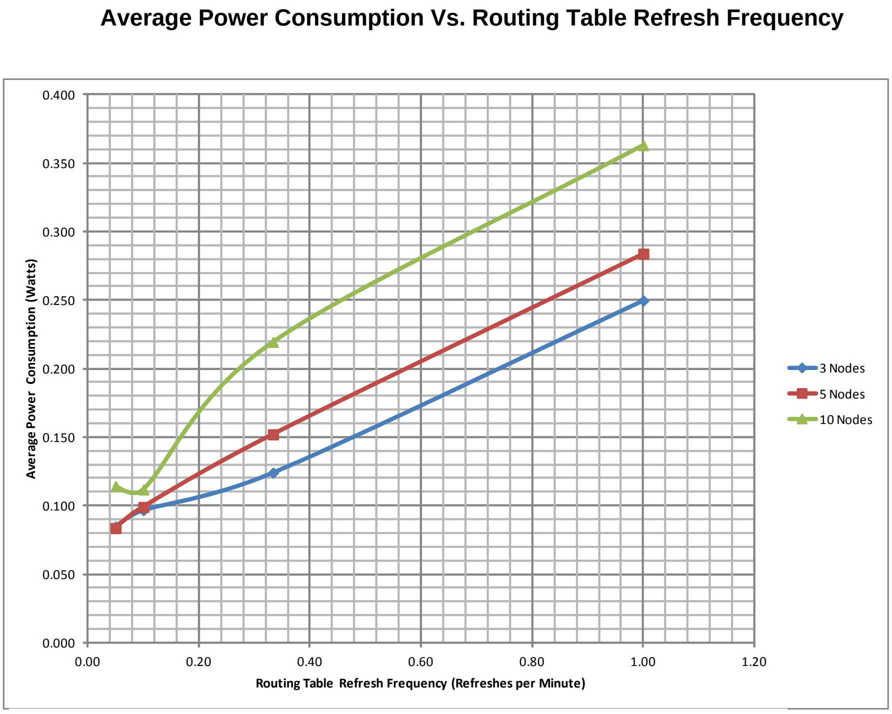
The practical configuration of the experiment was as discussed in section 6.3.1 and illustrated in Figure 6.1. The main variables for modification during this experiment were the number of participating nodes and a node's routing table refresh frequency. The refresh frequency was configured uniformly across all nodes for a single observation cycle. The refresh period (or delay between refreshes) was varied between 1, 3, 10 and 20 minutes. These values were selected to permit analysis of a wide dynamic range of possible values (i.e. from 3 to 60 times per hour). It was considered that if refresh frequency was to impact on energy consumption then a reduction factor of 20 would conceivably present a noticeable difference in observed data. Furthermore, the four refresh periods were considered plausible for a mobile P2P node operating scenario depending upon the level of churn experienced within the overlay network. Each re-configuration of the network with a new refresh period was monitored for a minimum of 180 minutes. In addition, the experiments conducted with a refresh period of 3 minutes and 20 minutes were monitored for up to 540 minutes to give the opportunity to assess any general longer term trends. No user data was entered into the P2P overlay network at any point during any of the tests. This experiment was focused solely upon routing table refresh frequency whose data was generated by the P2P overlay implementation logic. Acknowledgement was also made within the experiment configuration of the number of participating nodes, the other variable predominantly affecting routing table data. Tests were conducted with 3 nodes (all handsets), 5 nodes (all handsets) and 10 nodes (5 handsets and 5 emulators) at each of the 4 stated refresh periods. This test coverage of data allowed the results illustrated within Figures 6.2, 6.3 and 6.4 to be obtained. For reasons of presentation clarity, the variable data (y axis) is plotted in all graphs against the refresh frequency as number of refreshes per minute. A refresh delay period of 1 minute was considered a practical minimum to avoid interaction between refresh events that would distort results.

Experiment 1: Interpreting the Results

Interpreting the results data presented in these graphs leads to the following statements regarding the Bushfire P2P overlay behaviour:

i For all of the overlay network sizes investigated, the average power consumption across the range of handsets increases as the routing table refresh frequency increases (or alternatively the power consumption falls as the time between refresh events increases). It is clear that a factor of 20 reduction in the number of refreshes performed per hour, from 60 to 3 times, does have big impact on handset power consumption. Whilst approximately a linear relationship between power and frequency for the 5 node network configuration, the 3 and 10 node networks display some interesting variations away from this.

ii Whilst the trend indicates that reducing the refresh frequency also reduces the power consumption for a specific number of nodes, the inverse applies in relation to the number of participating nodes. For all refresh frequencies, the raw power consumption figure increases with the number of nodes. However, two anomalies in the data trends exist relating to the lower refresh frequencies. Namely there are almost similar



Refresh Frequency (Refreshes per Minute)	0.05	0.10	0.33	1.00
Delay Between Each Refresh (Minutes)	20	10	3	1
	↓	↓	↓	↓
Number of Nodes	Power Consumption (Watts)			
3	0.085	0.097	0.124	0.250
5	0.084	0.099	0.152	0.284
10	0.114	0.112	0.219	0.363

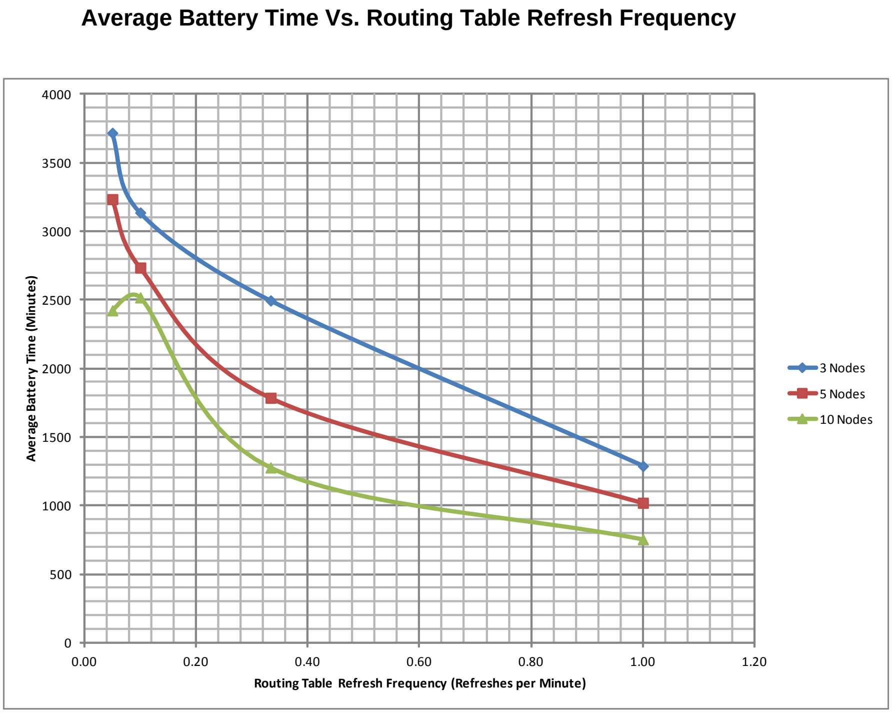
Figure 6.2: Average Power Vs. Routing Table Refresh Frequency

power consumptions for: 10 nodes, 10 minute and 20 minute refresh period; also for 20 minute refresh period, 3 and 5 nodes. Elaboration on theories for these aberrations appears later in the section.

iii There is an average power consumption difference of 40mW between handsets being in ‘standby’ state and participating in a 5 node overlay network. This figure does increase markedly with the doubling of node numbers from 5 to 10.

iv The trend for average processor activity follows a downward path as the refresh frequency drops, most likely contributing to the reduction in power consumption. The largest rate of decrease, for the two larger network sizes, is the initial switch from 1 minute to 3 minute refresh delay period. A reducing trend continues for the other readings taking into account some minor variation such as the convergence of reading for 5 and 10 nodes at the 20 minute refresh delay test. This perhaps may indicate that the extended time delay period itself has more significant impact than the number of participating nodes.

Considering the conducted experimentation further, whilst each graph shows different parameters, in fact each has an effect on the others. For example, the reduced power consumption results remarked upon in statement i. translate to an increase in average operating time for a node as the handset battery power source capacity is finite. There is also a link between lower processor activity (and refresh frequency) as highlighted in statement iv. that correlates with the reduced power consumption readings. A lower processor activity could be the result of power



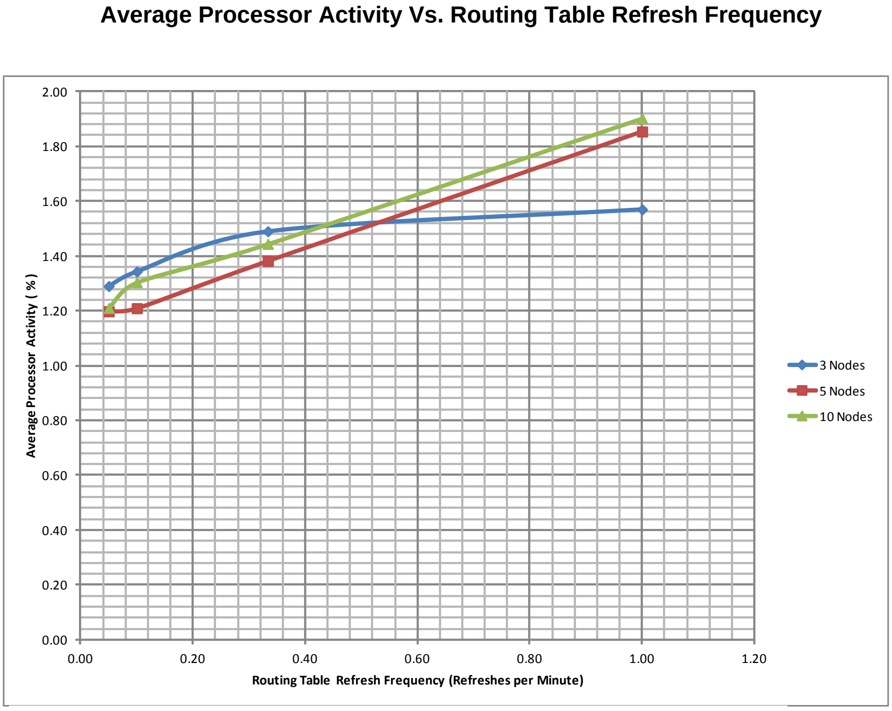
Refresh Frequency (Refreshes per Minute)	0.05	0.10	0.33	1.00
Delay Between Each Refresh (Minutes)	20	10	3	1
	↓	↓	↓	↓
Number of Nodes	Average Battery Time (Minutes)			
3	3718	3137	2495	1289
5	3234	2735	1786	1020
10	2423	2517	1277	754

Figure 6.3: Average Battery Time Vs. Routing Table Refresh Frequency

saving techniques employed both solely on the handsets and in conjunction with the operator GSM telephone network.

All the handsets utilised in the test configuration were configured with the power saving function activated. After a defined period (1 minute) of user inactivity (i.e. no keyboard interaction), the screen backlight was disabled and other power saving functionality instigated. In addition to this, a GSM power saving function called Discontinuous Reception (DRx) could be activated through coordination between the operator network and the transceiver control circuitry [38]. In essence, the approach involves turning the radio receiver off for a defined period of time (milliSeconds order of magnitude) and then re-activating for a synchronised short period to probe for any applicable messages from the network. This activity occurs autonomously, outwith the control or knowledge of the handset operator. For this reason, it is not directly attributable to the user settings for the Bushfire Framework. However, it should be included as a likely reason for some of the witnessed power consumption reduction, due to lower refresh frequencies being more likely to offer opportunity for this mode to be used.

Considering further the processor activity in relation to handset hardware, all the telephones utilised a device from the ARM 11 microprocessor series - a low power design component that has become extremely popular in



Refresh Frequency (Refreshes per Minute)	0.05	0.10	0.33	1.00
Delay Between Each Refresh (Minutes)	20	10	3	1
	↓	↓	↓	↓
Number of Nodes	Average Processor Activity (%)			
3	1.29	1.34	1.49	1.57
5	1.20	1.21	1.38	1.85
10	1.21	1.30	1.44	1.90

Figure 6.4: Average Processor Activity Vs. Routing Table Refresh Frequency

the mobile sector [138, 137]. All computational work done by the handset processor will result in power drawn from the battery. When not performing an operation (user or network defined) the microprocessor is capable of switching to a lower power standby state [1]. Reasonably assuming the Nokia handset designers made use of this microprocessor function, Figure 6.4 likely indicates that refreshing the routing table 60 times per hour (1 minute refresh delay) severely restricted the handsets opportunity to cease operational activities so that it could enter a standby state. Reducing the refresh frequency would have reduced the demand on the processor and thereby given more opportunity for power saving modes to be automatically employed. Furthermore, the effect of selecting a lower refresh frequency will not only impact on nodes making the queries but also those nodes answering the 'status request' messages, thus reducing energy consumption elsewhere in the overlay network. Again this autonomous aspect of the handset operation, although not solely attributable to the Bushfire Framework configuration, could be argued at least influences power conservation by establishing the correct environmental circumstances.

It is noted that the previous statements are broadly inline with expectations; in essence more participating nodes implies more entries in all routing tables which have to be maintained through more 'power consuming node queries'. Therefore whilst it is straightforward to accept that the base level power consumption will increase for more participating nodes in an overlay network, it is not so clear why the data shows the following 2 anomalies:

i With reference to Figure 6.2, the readings for 10 nodes, 20 minute refresh delay period (or lowest refresh frequency) shows an anomaly of a very slightly higher power consumption than the 10 nodes, 10 minute refresh delay period test.

ii With reference to Figure 6.2, data shows a 'convergence of curves' between 5 and 20 minute refresh delay period (i.e. a refresh event frequency reducing from 0.2 to 0.05 times per minute or 12 to 3 times per hour) applicable to all network sizes tested.

A possible explanation of these anomalies is that as the number of nodes increases this becomes more dominant than the refresh frequency. The recording of a slightly higher power consumption for 10 nodes, 20 minutes refresh delay period could indicate there is an 'optimum refresh frequency' between 0.05 and 0.16 (or 6.25 to 20 minute refresh delay period) regardless of participating node numbers. Obviously a lower optimum delay period (or higher refresh frequency) would have a positive effect on improving the routing table accuracy. This theory is supported by the fact that Kademlia has a maximum routing table size associated with it, meaning it becomes independent of overlay size for larger networks. This theory could only be investigated further with greater numbers of nodes participating in the overlay network in order to establish more substantial trend data.

6.3.3 Experiment 2: Energy Consumption Affected by Node Join Sequencing

As introduced in section 6.3.1, the focus of this experiment was to observe the impact of node routing table refresh sequencing within the overlay upon the energy consumption of a handset. This could be done in reference to either a node's initial registration in the network or its continued participation. For initial investigations discussed below, the experiment assumed a network of nodes was already in place so it was the sequencing of routing table refreshes that was under scrutiny. The motivation for conducting the test was to uncover further niche information relating to the correlation between handset energy consumption and ongoing routing table maintenance. The focus on refresh sequencing as a means to reduce energy consumption is through the principles of batch processing. Calder et al. [68] highlight in their work that there are 'significant energy savings' when using their batch processing recurrent software framework at the handset application level. As with experiment 1 (section 6.3.2), the impact of churn was not assessed within this experiment.

Experiment 2: Configuration and Results

The practical configuration of the experiment was as discussed in section 6.3.1 and illustrated in Figure 6.1. The main variable for modification during this experiment was the timing of a node's routing table refresh activity. 10 nodes were utilised (5 handsets and 5 emulators) and the refresh frequency was configured as 20 minutes across all nodes. The experiment consisted of 3 tests designed to highlight the impact of refresh sequencing, if there was one. For test 1, all 10 nodes were configured so that their routing table refresh happened at approximately the same time. (The practical limitations of performing the experiment resulted in a period of approximately 30 seconds across which all nodes joined the overlay.) Test 2 was a staggered approach where each of the 10 nodes was started at 5 minute intervals. For test 3, an approximation of random routing table refresh requests was investigated to mimic more natural P2P overlay behaviour. All tests ran for a period that would allow 4 routing table refresh actions to be completed on each node once the overlay network was fully established and stable. The measurements obtained for each of the 5 handset nodes in the 3 tests focused on an 80 minute window to allow a fair comparison of results. Figures 6.5, 6.6 and 6.7 visually illustrate the time intervals where the table refresh tasks were conducted in the tests.

As conducted in experiment 1 (section 6.3.2), measurements were recorded for an average of each handset for the experiment window and also averaged across all handsets to obtain a general trend result that was independent of hardware configuration. The results obtained from the 3 tests within experiment 2 are shown in table 6.2.

Experiment 2: Interpreting the Results

Interpreting the average measurements presented within table 6.2, purposeful staggering of refresh activities (test 2) on the peer nodes is not the best strategy to employ for handset energy reduction. More curiously the 'synchronised' (test 1) and 'quasi-random' (test 3) strategies had very similar average power consumptions which may be the result of coincidental behaviour or evidence of more repeatable results.

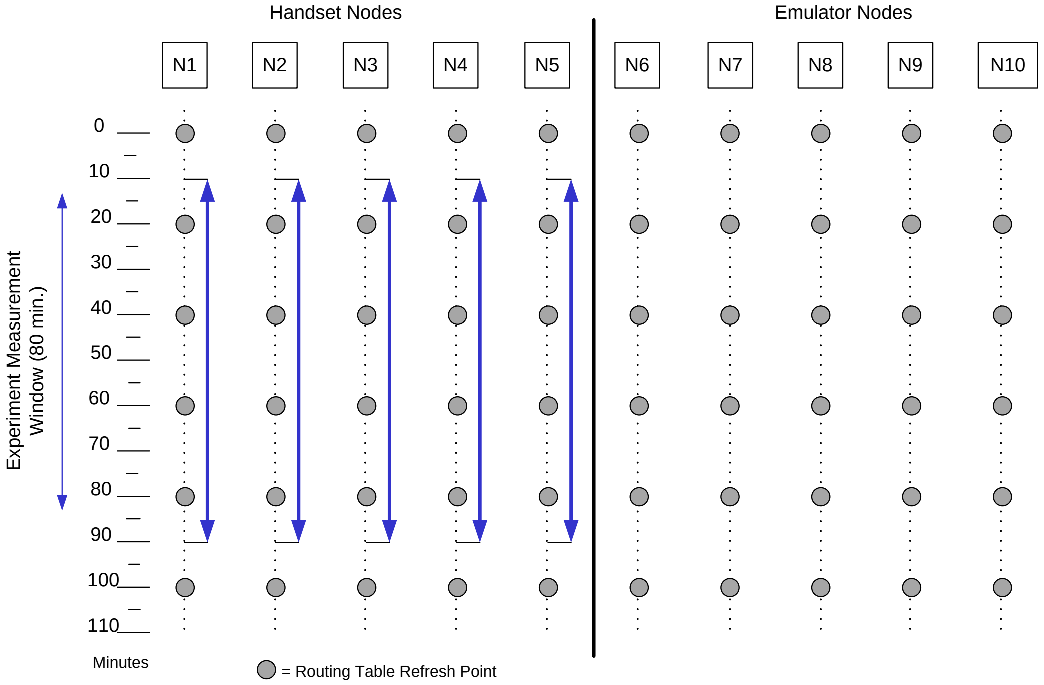


Figure 6.5: Experiment 2 - Test 1: Timing Diagram Illustrating a Synchronised Refresh of All Node Routing Tables Within an Overlay Network

Given the small number of participating nodes within this experiment, it is likely that all nodes were included in each other's routing tables and therefore a single refresh activity involved all nodes. For the staggered refresh approach (test 2) shown in Figure 6.6, this arrangement likely explains that none of the 10 nodes had an opportunity to 'rest' from a refresh activity and therefore consumed higher levels of energy throughout the whole 80 minute measurement period. Conversely, most likely for test 1 and also for test 3, there were potential opportunities for a node to reduce its level of refresh processing activity with associated power saving gains.

To conclude, more extensive research would be required to determine if there is any significant benefit to be gained from the software coding effort and commitment of handset resources required in implementing a synchronised routing table refresh policy. This would be in contrast to leaving nodes with the random approach which represents normal P2P overlay operation. Comparative benefits of the 'standard' random approach includes less complex routing table maintenance logic within each node.

6.4 Conclusions and Recommendations of an Energy Optimised Framework

Concisely described by Edwards [23] as a 'tug of war' between battery life and bandwidth provision, the results of experiments discussed in sections 6.3.2 and 6.3.3 reveal that this description remains valid for the Bushfire Framework P2P overlay. Focusing solely upon data and behaviour that is independent of the user has allowed analysis of parameters that when optimised will be applicable to all application scenarios of the framework. The following sections detail conclusions and recommendations for future work relating to energy consumption.

6.4.1 Conclusions

As was noted in the results of experiment 1 (section 6.3.2), the general trend was that power consumption by the handset was lower for a longer routing table refresh period. Some justification has been offered for this observation

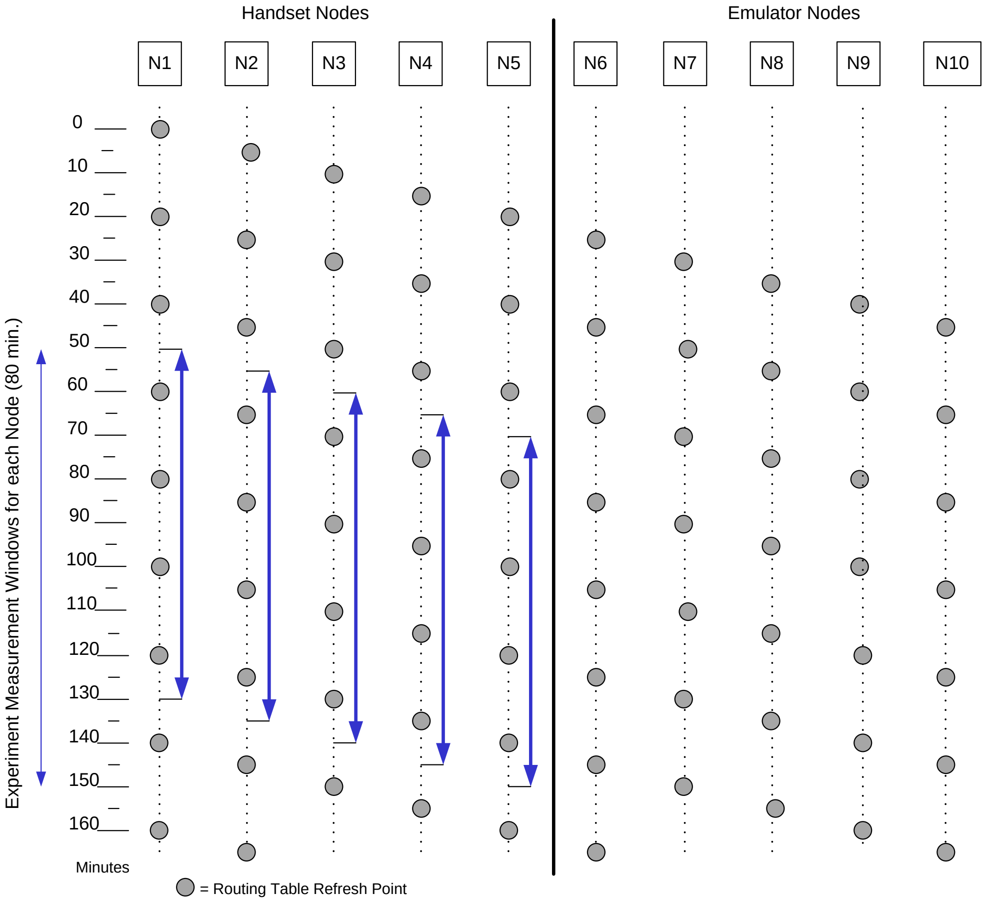


Figure 6.6: Experiment 2 - Test 2: Timing Diagram Illustrating a Staggered Refresh of All Node Routing Tables Within an Overlay Network

as well as a possible direction for further investigation in section 6.3.2. However, the power saving potential of this investigation may also be limited by another influencing factor; the operating mechanism employed SysProxy.

As discussed in section 4.4.6, SysProxy, in common with other ‘always connected’ applications, employs a ‘keep alive’ signalling method in order to maintain the TCP/IP network connection to the handset. The amount of telephone network signalling data traffic produced by an application is determined by the number of ‘connection states’ a handset will transition through in supporting the flow of application data. Connection states are an intrinsic part of the telephone network operating design and a handset will always be in a particular state when registered on a network. Each state transition for a handset consumes energy through the use of the radio transceiver and can result in 1 to 3 seconds of signalling effort. Estimates of current consumption for different states vary, but with a difference of 195mA given between the lowest and highest values in one report [78] it can be understood the significant contribution this control traffic switching has on battery charge depletion. In essence, there is a ‘base level’ of energy consumption within a handset that cannot be reduced. The optimisation techniques discussed in this chapter would only lower energy consumption to a defined minimum. With reference to tests done in this report [78], it eloquently quantifies the direct impact that polling techniques such as ’keep alive’ signals make on the handset energy resources: ‘An application that sends one ’keep alive’ message every minute uses the same battery power in only eight hours as keeping the handsets back-light on for a full hour.’

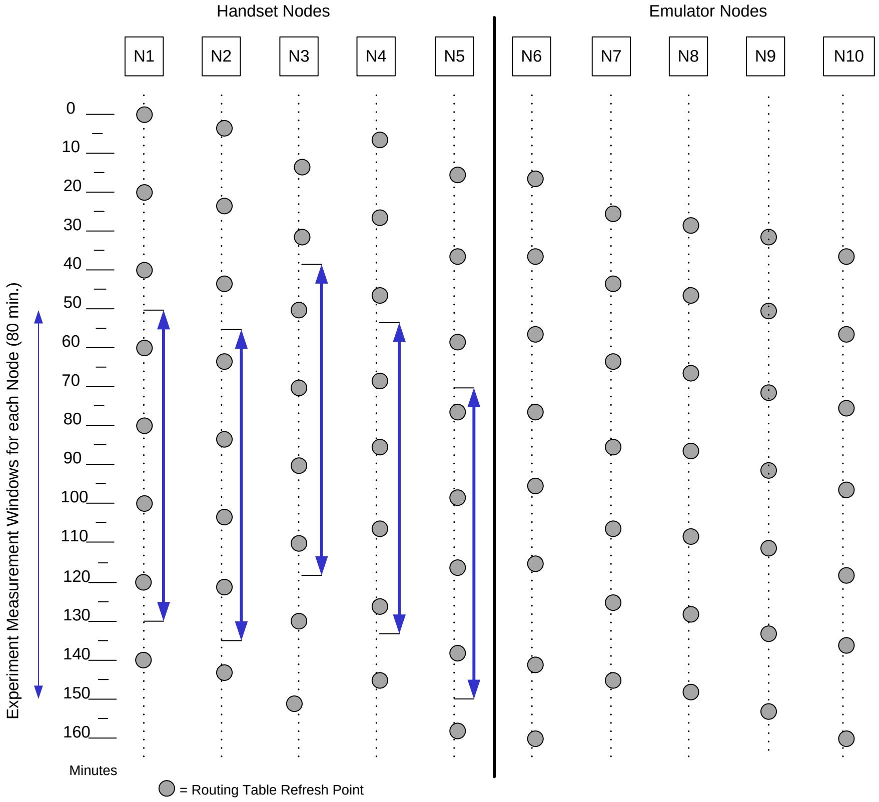
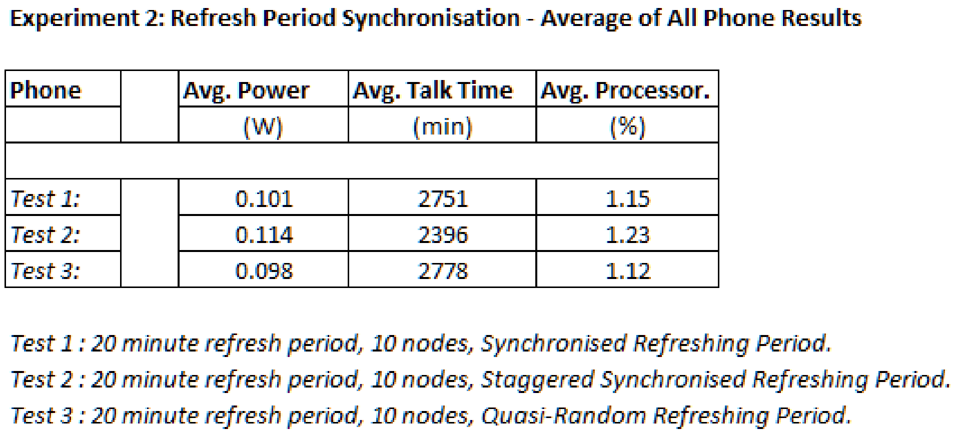


Figure 6.7: Experiment 2 - Test 3: Timing Diagram Illustrating an Approximated Random Refresh of All Node Routing Tables Within an Overlay Network

The alternative viewpoint on the limitations imposed by using ‘constantly maintained’ handset network connections is that effort will now focus on reducing the power consumption in other, more intuitive, ways. Since 2008, there has been an understood power budget within Nokia for mobile handsets of 3 Watts total with a maximum allocation of 0.6 Watts for the application processor [55, 5]. This limit has also been considered more recently in relation to Android OS handsets [134], indicating a commonality of constraints across platforms. This 0.6 Watt power budget would cover all applications running on the processor. Minimising the consumption associated with the P2P overlay would give more overhead to the application(s) running on top of the framework. Another example method of reducing power consumption was investigated in experiment 2 (section 6.3.3). Indicative results were obtained that a coordination of node table refresh activities may possibly save energy across all participating handset nodes. It is acknowledged that experiment 2 was highly simplified and further investigation would be required. Furthermore a mechanism would be needed for wide scale synchronisation, the network operator or GPS time signal being obvious candidates. However, implications of this alternative approach to routing table maintenance would need to be fully considered as the network traffic would ‘spike and dip’ due to the synchronous actions (unlike within a standard P2P network operating with ‘quasi-random’ routing table refresh activities).

6.4.2 Recommendations

Recommendations to extend the performance evaluation could take two approaches: minor enhancements to the existing framework design in order to optimise performance or a re-evaluation of the ‘design trade-off’ that exists when mobile telephone and P2P parameters combine. Considering enhancements, experiment 1 (section 6.3.2) could be extended by investigating the optimisation of data transfer necessary for a routing table refresh. This further work could be performed using networking tools such as Wireshark [125] and a Nokia S60 platform variant IPTrace [46] to conduct data bandwidth measurements in unison with modification of the framework routing table



Experiment 2: Refresh Period Synchronisation - Average of All Phone Results

Phone		Avg. Power	Avg. Talk Time	Avg. Processor.
		(W)	(min)	(%)
Test 1:		0.101	2751	1.15
Test 2:		0.114	2396	1.23
Test 3:		0.098	2778	1.12

Test 1 : 20 minute refresh period, 10 nodes, Synchronised Refreshing Period.
Test 2 : 20 minute refresh period, 10 nodes, Staggered Synchronised Refreshing Period.
Test 3 : 20 minute refresh period, 10 nodes, Quasi-Random Refreshing Period.

Table 6.2: Telephone Handset [152, 153, 154, 155, 156] Measured Data for Investigating the Impact of Routing Table Refresh Synchronisation Upon Energy Consumption

data protocol. Experiment 2 could be extended by full integration of the functionality that was rudimentarily tested in section 6.3.3 if it were deemed of sufficient benefit.

Examining other aspects of research related to the experiments conducted introduces a compromise to be determined between desired routing table accuracy, refresh frequency and node churn. In essence, fixing a desired level of routing table accuracy will depend upon a refresh frequency that is linked to the observed level of node churn. According to Stutzbach et al. [97] with a study of churn across different classes of P2P systems 'peer inter-arrival times follow a Poisson distribution'. The study also observed that in their networks 'a large portion of participating peers at any point of time are highly stable while the remaining peers turnover very quickly'. These findings give an indication that it may be possible to characterise churn within a 'typical mobile scenario'. More accurately predicting churn would allow the other parameters to be optimised. To determine optimum refresh frequency would need churn data from the PC P2P network as well as connectivity data for the 'average' mobile user. Connectivity data is important, as unlike PC-based networks, a user's mobile telephone is subject to more external usage influences. For example, making/receiving telephone calls (which disconnects the data channel), the changing physical environment and being in constant close proximity to the user all contribute to varying levels of churn. Academic background research such as reports [57] and extended surveys (such as a 'Mobile Life Study' conducted by TNS [129] with 34,000 people across 43 countries) help to illuminate usage habits in more detail. However questions on what influences user churn in the mobile sector, such as those listed below, may only be answerable with data from mobile telephone operators or P2P network development teams.

- Does user age influence churn? (e.g. influence on daily / weekly usage patterns such as school, work, etc.)
- Does a majority or minority of users turn their phone off at night? (e.g. possible impact on content availability in a network unless users are located worldwide)
- Do daily work patterns affect connectivity on a large scale? (i.e. most people work Monday to Friday in a town / city where coverage is generally better)
- Do seasonal variations affect anything? (eg. national holiday plans impacting on usage)
- Will the negative impact on battery life promote 'free riding' behaviour? (i.e. Where users turn on an application only to get the data they need.) Studies have indicated 'free riding' has been as high as 85% on standard P2P networks (Gnutella) where resources are less constrained [42]. Can this conduct be counteracted by promoting 'stable node' behaviour through either rewarding longer or punishing shorter connection times?

The sections within this chapter have presented many considerations on the subject of energy usage optimisation. However from emerged themes, it is clear that scope exists for further research to aim for a ‘longer than a day’ target charge cycle (even if it appears to be beyond the current hardware technology). Recent statistics predict that smartphones are expected to rise to a 45% contribution of the total mobile telephone market shipment in 2015 [43]. Therefore it could be assumed that technology will likely improve and energy usage optimisation will remain active as a research topic.

Chapter 7

Conclusions and Further Work

This chapter discusses the technologies, results and verdicts presented in this thesis to present a concise summary of the Bushfire P2P Framework. The conclusions drawn from this work include highlighting its positive attributes and those aspects requiring further investigation.

7.1 Achievements of the Approach

This thesis has presented a software development framework as a means to incorporate P2P applications into handsets operating on mobile telephone networks. Smartphones have reached both a level of user acceptance and technical capability that the use of P2P-oriented handset applications has become feasible. The system (code-named Bushfire) was constructed in two constituent parts comprising a handset component and a proxy server (code-named SysProxy). Established techniques for both aspects were combined and optimised in a novel approach to overcome technical constraints imposed by the telephone networks. To promote adoption with third party applications ease of integration was prioritised (through a straightforward API) and the applicability of the underlying transport protocol was modified and broadened.

Throughout the research process, design decisions have been required to bring together a P2P framework system that uses established techniques to form an innovative approach to overcoming notable obstacles. The selection of P2P overlay was derived from a thorough process to evaluate available options (refer to sections 3.1 and 3.2) resulting in the choice of Kademlia as a suitable protocol meeting all of the desired criteria. An existing implementation of Kademlia for the chosen development mobile platform, Nokia / Symbian S60, was selected for evaluation, dissection, enhancement and ultimately integration with the other integral component of the system, SysProxy. In order to overcome networking constraints imposed by the operator networks, a proxy server, dedicated to the Bushfire P2P traffic, performs NAT traversal function. Location on a computer with a public internet IP address is the only requirement for this application to allow communication between participating handset nodes. The resulting architecture of combining these two systems is the logical ring of a Kademlia P2P system running atop a star topology with Sysproxy at the centre (section 4.3). Chapter 6 has focused upon discussion of optimising performance; for P2P on mobile devices this was targeted at power consumption as it ultimately dictates node operating time. Experimentation discussed in section 6.3 gave direction on configuration for the developed system and also for further research. Whilst this research did achieve many positive outcomes from combining and enhancing current technological approaches in P2P overlay networks and NAT traversal, there were also limitations and opportunities for further research (as discussed in sections 7.3 and 7.4 respectively).

7.2 Aligning Aims with Achievements

Mapping the aims of this work as detailed in section 1.2 to the achievements attained produces the following account:

- **Provide an appropriate P2P framework that operates on 'smartphone' handsets using an internet connection across the mobile telephone network.**

Chapters 3 and 4 present the technology decisions, optimisations and design architecture (section 4.3) for constructing a P2P framework that operates on the mobile telephone network. The system was constructed in two constituent parts: the handset component (section 4.4.5) to interface with third party applications and a proxy server (section 4.4.6) to implement NAT traversal of the encountered networks.

- **Provide an appropriate P2P framework within which third party applications could be easily developed using a clearly defined application programming interface (API).**

  Chapters 3 and 5 discuss the review, evaluation and design process employed in selecting, implementing, modifying and optimising a suitable P2P technology for the framework. Elements of the API created and adapted for the framework are discussed in sections 4.4.2 and 4.4.3. Chapter 5 presents the applications that have been developed to either help test (Bushfire Test Engine, section 5.1) or work with the Bushfire P2P framework (Buddy Net, section 5.2; SupportNet, section 5.3).

- **Provide observational data, analysis and guidance on operating behaviour both for the implementation discussed within this thesis and for further research effort.**

  Elements of chapters 4 and 5 in addition to the entire contents of chapter 6 are dedicated to the analysis and evaluation of the Bushfire framework. Chapter 6 in particular offers guidance on the subject of energy optimisation, a topic of importance when operating mobile devices. This is subject is explored deeper through experimentation (sections 6.3.1 to 6.3.3) investigating a configuration parameter and operating sequence connected with the Bushfire framework to seek optimum power consumption.

To summarise, all of the original aims of this thesis have been met by the work produced.

7.3 Limitations of this Work

Given the varied technical environment within which this solution had to operate, there were a number of limitations in the scope and implementation as well as assumptions made regarding the research conducted. A list of the most significant limitations is presented below:

- System architecture: The star topology configuration of SysProxy represents a potential weakness in the design (although counter arguments have been presented in section 4.5). Although SysProxy does not store any network data, its presence to route data between nodes is essential and therefore it presents a restriction point for data flow and a significant weakness in the resilience of the complete system.

- Limitation with design of SysProxy application: Currently only a finite number of supported users (14535) can be supported through a single SysProxy application (refer to section 4.4.6). The IMEI recording mechanism employed by SysProxy to help reduce the impact of network disconnects prevents this number being viewed as ‘simultaneous users’. This will be a significant limitation to achieving large scale adoption that aids P2P overlay operation. Alternative strategies such as multiple, linked SysProxy applications would be essential to achieve node numbers equivalent in size to the desktop computer overlay counterparts.

- Limitations with knowledge of mobile networks: Assumed connectivity variations amongst operators have attributed to differences in operating practices and NAT / firewall techniques. The current design has no NAT traversal optimisation techniques incorporated due to lack of accurate operator information.

- Limitations of P2P systems that are ‘amplified’ within the mobile sector: Establishing a network with enough nodes to become self sustaining can be challenging when there are so many potential hardware platforms to support. The practical work produced for this thesis focused on the Symbian / Nokia S60 platform. Whilst acknowledged as a popular smartphone OS, any wide scale adoption of this P2P framework would require porting the code across to (potentially many) other mobile platforms. Node churn and data longevity within the overlay are two other key parameters that impact the effectiveness of the mobile P2P network.

- Limitations with mobile handset technology: Current practical working constraints meant that units under test could only be operated for periods under 12 hours when operating on battery power. To run tests longer than this, connection to a power supply was required - but then mobility was obviously compromised.

In addition to those limitations listed above, section 6.3.1 discusses the drawbacks and reservations acknowledged with the energy optimisation experiments. The most fundamental of these could be summarised as difficulty in achieving total control in a realistic experiment scenario. This was primarily due to many influencing variables being beyond manual manipulation such as telephone network operating parameters. However, the knowledge gained from trend data, even with these limitations, has been useful in understanding the operation of the Bushfire framework.

7.4 Further Work

Due in part to the limitations indicated in the previous section (7.3), there are a number of opportunities where the current research could be extended with further work. A list of the most noteworthy suggestions for future research is shown below:

- Continuation of energy optimisation experimentation themes as indicated in chapter 6. Research focus could be targeted towards processor effort or bandwidth (data consumption) optimisation on the handset as both would have an effect on power consumption and therefore the node participation times.

- Improved design for the SysProxy application. A re-design could remove its perceived weakness of a low participating node limit by coordinating the activity of multiple SysProxy servers. For example an additional hierarchy layer could be formed of a 'ring of SysProxy servers'. Assuming the SysProxy application location at the centre of the network topology is understood as a bottleneck to data flow, then this design modification would alleviate any concerns as well as increasing the network resilience. Furthermore, through the addition of data packet inspection functionality into SysProxy, message routing priority or access control mechanisms could be implemented within the Bushfire framework P2P overlay.

- With the hierarchical segmentation of the SysProxy network, further implementation decisions could be taken to aid likely routing paths between mobile nodes. SysProxy servers could be arranged to minimise 'inter-SysProxy' communication and thereby minimise message latency. For example mobile nodes could be allocated to a SysProxy server based upon their physical location (such as a city or country) or a conceptual location (such as mobile operator network or work organisation). The multi-SysProxy approach could be extended in another direction by transferring the server functionality onto selected mobile nodes themselves. This approach would remove the need for separate physical server infrastructure but assumes that the mobile node has a public IP address, spare processing capacity, constant power source and a high bandwidth internet connection. This is a demanding set of assumptions that are unlikely for a truly mobile node but feasible for a scenario where a handset is charging overnight at a user's home. In this instance, 'sleeping' mobile nodes in one world time zone could provide the SysProxy routing resources for 'awake and mobile' nodes in a different time zone. Less demand could also be put on the handset resource by reducing the number of nodes it acts as a SysProxy server for.

- Further research into the NAT traversal techniques employed within the Bushfire framework and those NAT systems used in the mobile telephone networks. A line of research under consideration is to reduce the workload on the SysProxy application by modifying its behaviour from a proxy server to a communications 'broker'. SysProxy could be used initially to establish direct connections between two nodes and then remove itself from the communications path letting the nodes communicate directly. Two previously published techniques would be used with this approach: NatTrav [24] for brokerage services logic and Hole Punching for the direct communications link [52]. Work by Haddad [37] in combining these techniques would be a suitable starting point for this research topic. Furthermore, is the (eventual) change to an IPV6 address structure for mobile devices likely to bring benefits to the Bushfire framework or cause problems? The official exhaustion of available IPV4 addresses during early 2011 will ensure IPV6 compatibility and usage become increasingly prevalent within all future internet connected devices [146]. This subject represents an area of future research for reasons of interoperability between IPV4 and IPV6 equipment as currently the two versions only interact via specialist translation mechanisms. Ultimately this aspect of future research is desirable to promote mass future adoption through more efficient integration.

- Extension of the deployed test (or beta) network to include much larger numbers of nodes. A wide scale deployment exercise would yield much more useful data about the practical usage of P2P overlays within the mobile domain. In order for this to happen it needs to be application or content driven so that users both have the desire to use it and it is under a realistic scenario. In the first instance, a suggested target audience could be conference attendees (e.g. 60,000 attended the 2011 Mobile World Congress [87]) who offer a constrained test environment / timetable and are likely to be receptive if there is an incentive.

- To achieve the above action, it is likely that the developed code will have to be ported to other hardware platforms to increase probability of adoption. The most obvious first candidate for software coding effort would be the Android OS platform as it has a supported development environment and is available on a wide choice of handsets. Furthermore predicted adoption rates for Android based devices look healthy as they were calculated at approximately 15 million units per quarter worldwide in 2010 [2].
- Further research is needed into security related matters applicable to the Bushfire Framework and those applications that would interface with it. Building upon the suggestions given in sections 4.4.7 and 5.4, it would be necessary to implement a minimal set of functionality within the framework prior to any extensive deployment test (as discussed above). This would meet the user's likely expectations for a mobile handset networked application whilst minimising the risk of incurring any undesirable consequences as a result of a user data privacy breach. A minimal functionality set could be defined as meeting the criteria laid out in section 4.4.7: ensure secure communication (using encrypted data packets), know your peers (by requiring user registration or logging a phone number) and access control (using password protection).

Considering the first list item above, energy consumption relating to P2P network operation is a parameter that has been seriously considered only with respect to the mobile domain. In an era where the environmental impact of every action is to be considered, optimising traditional P2P networks which are typically installed upon desktop computers could yield significant power savings when scaled up across many thousands of users.

7.5 Summary

This chapter has highlighted the primary aspects of the research conducted for this thesis. Implementation of P2P derived systems has become a mainstream technology in the traditional computing sector. The system produced for this thesis employs a novel approach to translate this across to the mobile telephone domain. Key developments included producing an optimised handset application framework (creating a Kademlia protocol based P2P network) working with a specialised proxy server to convey messages across multiple network boundaries (NAT traversal). The crucial advantage of this framework approach is to permit third-party applications to function on the restricted mobile telephone network with a P2P network as the underlying data transport / storage infrastructure. This, coupled with a simple API, lowers the barrier to feasibility for an application as it removes the requirement for centralised 'server style' infrastructure (along with its associated costs and maintenance overheads).

Given the findings reported from this work, there is an opportunity for handset applications to be developed to meet real use cases or further research to be conducted for a growing commercial sector where mobile application downloads in general are predicted to reach approximately 48 billion in 2015 [43].

References

[1] ARM. Application note 143: Understanding ARM 11 processor power saving modes. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dai0143c/index.html, 2007. Accessed May 2011.

[2] C. Arthur. Google 'activating 160,000 android phones a day'. http://www.guardian.co.uk/technology/2010/jun/23/google-android-eric-schmidt, 2010. Accessed June 2011.

[3] C. Arthur. Why file-sharing has killed 'unlimited' mobile data contracts. http://www.guardian.co.uk/technology/2010/jun/11/mobile-data-unlimited-end, 2010. Accessed June 2010.

[4] G. Association. 2008 GSMA corporate brochure. http://www.gsmworld.com/documents/gsm_brochure.pdf, 2008. Accessed March 2010.

[5] G. Bosch and M. Kuulusa. Nokia webinar: See the power of software - optimizing mobile applications. http://www.developer.nokia.com/Resources/Multimedia/Webinars.xhtml, 2009. Accessed January 2011.

[6] J. Buford. Management of peer-to-peer overlays. In *International Journal of Internet Protocol Technology, Vol. 3, No. 1*, 2008.

[7] P. Burtwistle, M. Kolberg, M. Wilson, M. Blunn, and E. Magill. A framework for mobile applications based on a structured P2P overlay, Jan. 2009. 6th IEEE Consumer Communications & Networking Conference CCNC 2009 Demonstration Poster.

[8] H. Charaf, P. Ekler, and I. Kelenyi. Bittorrent at mobile phones. In *5th IEEE Consumer Communications & Networking Conference (CCNC '08)*, 2008.

[9] M. Collier. Smartphones vs. personal computers: The battle is on. http://finance.yahoo.com/news/Smartphones-vs-Personal-iw-4041552039.html?x=0&.v=1, 2010. Accessed March 2011.

[10] G. Corporation. Google mobile maps. http://www.google.co.uk/mobile/maps/. Accessed March 2010.

[11] N. Corporation. Nokia energy profiler. http://www.forum.nokia.com/info/sw.nokia.com/id/324866e9-0460-4fa4-ac53-01f0c392d40f/Nokia_Energy_Profiler.html. Accessed February 2011.

[12] N. Corporation. Nokia mobile maps. http://www.nokia.co.uk/services-and-apps/ovi-maps/main. Accessed March 2010.

[13] N. Corporation. S60 platform: Effective power and resource management - S60 platform guide (ver. 3.1.), 2007. Accessed December 2009.

[14] N. Corporation. Nokia completes the transaction to outsource its symbian software development and support activities to accenture. http://press.nokia.com/2011/09/30/nokia-completes-the-transaction-to-outsource-its-symbian-software_-development-and-support-activities-to-accenture/, 2011. Accessed October 2011.

[15] S. Corporation. Peer-to-peer file sharing - the impact of file sharing on service provider networks. downloads.lightreading.com/wplib/sandvine/P2P.pdf, 2002. Accessed March 2011.

[16] S. Corporation. Analysis of traffic demographics in broadband networks. In *IETF Workshop on Peer to Peer Infrastructure*, 2008.

[17] J. Curran. Notice of internet protocol version 4 (IPv4) address depletion. https://www.arin.net/knowledge/about_resources/ceo_letter.pdf, 2009. Accessed May 2010.

[18] F. Dabek, I. Stoica, R. Morris, D. Liben-Nowell, H. Balakrishnan, M. Kaashoek, and D. Karger. Chord: A scalable peer-to-peer lookup protocol for internet applications. In *ACM SIGCOMM '01*, 2001.

[19] Z. Despotovic, J. W. Lee, H. Schulzrinne, and W. Kellerer. 0 to 10k in 20 seconds: Bootstrapping large-scale DHT networks. In *IEEE International Conference on Communications (ICC '11)*, 2011.

[20] R. Deters and N. Ting. 3LS - a peer-to-peer network simulator. In *3rd IEEE International Conference on Peer-to-Peer Computing (P2P '03)*, 2003.

[21] K. Dezhgosha and S. Kirkman. Security review of P2P applications and networks. In *International Conference on Internet Computing*, 2011.

[22] C. Edwards. Getting more power to the iphone. http://www.guardian.co.uk/technology/2008/aug/21/apple.iphone, 2008. Accessed March 2011.

[23] C. Edwards. Tug of war as app's take off, 2010. Article November Edition.

[24] J. Eppinger. TCP connections for P2P apps: A software approach to solving the NAT problem. http://reports-archive.adm.cs.cmu.edu/anon/isri2005/CMU-ISRI-05-104.pdf, 2005. Accessed June 2011.

[25] P. Eronen, H. Haverinen, and J. Siren. Energy consumption of always-on applications in WCDMA networks. In *65th IEEE Vehicular Technology Conference*, 2007.

[26] L. Feeney. Introduction to MANET routing. http://www.nada.kth.se/kurser/kth/2D1490/05/lectures/feeney_mobile_adhoc_routing.pdf, 2005. Accessed November 2011.

[27] m. I. E. T. Force. IETF working group: Multiparty multimedia session control (mmusic). http://datatracker.ietf.org/wg/mmusic/charter/, 2010. Accessed May 2010.

[28] B. Forstner and I. Kelenyi. Distributed hash table on mobile phones. In *5th IEEE Consumer Communications & Networking Conference (CCNC '08)*, 2008.

[29] N. Forum. How well do you know about symbian S60 3rd edition? http://www.nokiausers.net/forum/symbian-s60v3/17627-how-well-do-you-know-about-symbian-s60-3rd-edition.html, 2007. Accessed June 2010.

[30] N. Forum. Mobile internet battery life: Challenges and solutions. http://www.forum.nokia.com/info/sw.nokia.com/id/48be6ccb-aded-4ebf-9779-97d58fd042dc/mobile_battery_life_day1_pt1.html, 2009. Accessed December 2009.

[31] N. Forum. Carbide.c++. http://www.forum.nokia.com/Library/Tools_and_downloads/Other/Carbide.c++/, 2011. Accessed March 2011.

[32] S. Freak. Symbian and ARM cooperate in bringing symetric multi-processing (SMP). http://www.symbian-freak.com/news/007/10/symbian_and_arm_cooperate.htm, 2007. Accessed June 2010.

[33] N. Ganguly. Self regulated search in unstructured peer to peer networks. http://www.facweb.iitkgp.ernet.in/~niloy/PRESENTATION/IBM%20day%20talk%20April%2028%202007.ppt, Apr. 2007. Accessed February 2010.

[34] I. GmbH. Ipoque internet study 2008/09. http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009, 2009. Accessed February 2010.

[35] S. R. Group. Smartphones and a 3G network - SRG webinar. http://www.signalsresearch.com/Docs/SRG%20Webinar%20Presentation%20II.pdf, 2010. Accessed March 2011.

[36] S. R. Group. The trouble with twitters report preview. http://www.signalsresearch.com/Docs/SA%20012810%20Smartphone%20Signaling%20Preview.pdf, 2010. Accessed March 2011.

[37] Z. Haddad. Implementing a TCP hole punching NAT traversal solution for P2P applications using netty. University of Stirling Masters Thesis, 2010. Accessed June 2011.

[38] L. Harte. Introduction to GSM - definition: Discontinous reception. http://www.wirelessdictionary.com/Wireless-Dictionary-Discontinuous-Reception-DRx-Definition.html, 2009. Accessed May 2011.

[39] M. Hauswirth and K. Aberer. An overview on peer-to-peer information systems. In *4th International Meeting Distributed Data & Structures (WDAS '02)*, 2002.

[40] G. Heiser and A. Carroll. An analysis of power consumption in a smartphone. In *USENIX annual technical conference (USENIXATC '10)*, 2010.

[41] T. Henderson, F. B. Abdesslem, and A. Phillips. Less is more: Energy-efficient mobile sensing with senseless. In *MobiHeld '09*, 2009.

[42] D. Hughes, G. Coulson, and J. Walkerdine. Free riding on gnutella revisited: The bell tolls? IEEE Distributed Systems Online, Vol6. No.6, 2005. Accessed June 2011.

[43] In-Stat. Mobile application downloads to approach 48 billion in 2015. http://www.instat.com/newmk.asp?ID=3156\&SourceID=00000652000000000000, 2011. Accessed June 2011.

[44] A. Inc. iphone - battery/usage suggestions. http://www.apple.com/batteries/iphone.html, 2011. Accessed March 2011.

[45] E. N. Inc. Eyeball: Firewall and NAT traversal server. http://www.anyfirewallserver.com/images/eyeballanyfirewallserver.pdf, 2010. Accessed May 2010.

[46] Inmote. IP trace for S60. http://www.inmote.nl/site/content/iptrace-s60. Accessed October 2010.

[47] D. International. Digi connect application guide cellular IP connections (uncovered). http://www.digi.com/pdf/appguide_connectcellular_ipconsiderations.pdf, 2005. Accessed March 2011.

[48] D. International. Remote wireless networking 101. www.digi.com/pdf/book_remotewireless101.pdf, 2006. Accessed March 2011.

[49] R. Jain, S. Guha, and N. Daswani. An experimental study of the skype peer-to-peer VoIP system. In *International Workshop on P2P Systems (IPTPS)*, 2006.

[50] J. Jetcheva, J. Borch, D. Maltz, D. Jognson, and Y. Hu. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *4th ACM/IEEE International Conference on Mobile Computing and Networking*, 1998.

[51] R. Jiminez. Kademlia on the open internet: How to achieve sub-second lookups in a multimillion-node DHT overlay. Licentiate Thesis, Royal Institute of Technology, Stockholm, Sweden, 2011. Accessed October 2011.

[52] D. Kegel, B. Ford, and P. Srisuresh. Peer-to-peer communication across network address translators. In *USENIX Annual Technical Conference (USENIXATC '05)*, 2005.

[53] I. Kelenyi. Mobile kademlia for symbian OS. http://www.aut.bme.hu/Portal/MobileDHT.aspx?lang=en, 2008. Accessed October 2008.

[54] J. Kujala. Battery life optimization: Forum nokias energy saving tips webinar series. http://www.forum.nokia.com/Library/Multimedia/Webinars.xhtml, 2009. Accessed December 2009.

[55] K. Kuusilinna. DATE 2008 workshop: The mobile fight - software versus power. http://www.imec.be/SE4ES/DATE08/DATE08-files/DATE08-SE4ES-Nokia-Kuusilinna.pdf, 2008. Accessed January 2011.

[56] LastFM. LastFM mobbler. http://code.google.com/p/mobbler/. Accessed March 2010.

[57] Z. Laughlin. Mobile phone users: A small-scale observational study. http://www.aber.ac.uk/media/Students/zgl9901.html, 2001. Accessed May 2011.

[58] J. Lendino. Smartphone 101: A look at the past, present & future. http://pctoday.com/Editorial/article.asp?article=articles/2006/t0402/12t02/12t02.asp&guid=, 2006. Accessed March 2010.

[59] E. Leonardi, G. Carofiglio, C. Chiasserini, and M. Garetto. Route stability in MANETs under the random direction mobility model. *IEEE Transactions on Mobile Computing*, 8:1167–1179, 2009.

[60] J. Li, J. Stribling, R. Morris, and M. F. Kaashoek. Bandwidth-efficient management of DHT routing tables. In *2nd Symposium on Networked Systems Design and Implementation*, 2005.

[61] J. Li, Y. Xiao, H. Li, and Y. Pan. On optimizing energy consumption for mobile handsets. In *IEEE Transactions on Vehicular Technology, Vol. 53 No. 6*, 2004.

[62] S. Lim, E. Lua, J. Crowcroft, M. Pias, and R. Sharma. A survey and comparison of peer-to-peer overlay network schemes. In *IEEE Communications Surveys & Tutorials*, 2004.

[63] H. G. U. Ltd. Three network coverage. www.three.co.uk/Coverage, 2010. Accessed March 2010.

[64] T. T. Ltd. TATA DOCOMO buddynet. http://www.tatadocomo.com/buddynet.aspx. Accessed November 2010.

[65] E. K. Lua, J. Buford, and H. Yu. *P2P Networking and Applications*. Morgan Kaufman / Elsvier, 2009.

[66] R. Mahy, J. Rosenburg, J. Weinberger, and C. Huitema. The internet society RFC3489: STUN - simple traversal of user datagram protocol (UDP) through network address translators (NATs). http://tools.ietf.org/html/rfc3489, 2003. Accessed May 2010.

[67] R. Manduchi, C. Margi, and K. Obraczka. Characterizing system level energy consumption in mobile computing platforms. In *IEEE International Conference on Wireless Networks, Communications and Mobile Computing*, 2005.

[68] M. Marina and M. Calder. Batch scheduling of recurrent applications for energy savings on mobile phones. In *IEEE Secon*, 2010.

[69] G. Marsden and M. Jones. *Mobile Interaction Design*. Wiley, 2006.

[70] P. Matthews, J. Rosenburg, and R. Mahy. The internet engineering task force RFC5766: TURN - traversal using relays around NAT. http://tools.ietf.org/html/rfc5766, 2010. Accessed May 2010.

[71] D. Mazieres and P. Maymounkov. Kademlia: A peer-to-peer information system based on the XOR metric. In *1st International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[72] L. Meng, B. Wen, and F. Liu. *A Novel Integrated Supporting System for Mesh-Pull Based P2P IPTV*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2008.

[73] R. Menta. Top P2P applications: 1.6 million PCs rank them. http://www.mp3newswire.net/stories/8002/P2P.html, 2008. Accessed March 2011.

[74] T. Middle East Technical University, Ankara. Lecture notes on fundamentals of design. http://www.eee.metu.edu.tr/~design/lecture_notes/DesignProcessVer171007.ppt. Accessed May 2010.

[75] M. Miller. *Discovering P2P*. Sybex Inc., 2001.

[76] MIT. P2P sim. http://pdos.csail.mit.edu/p2psim/index.html. Accessed March 2011.

[77] M. V. R. Murthy. Mobile based primary health care system for rural india. http://www.w3.org/2008/02/MS4D_WS/papers/cdac-mobile-healthcare-paper.pdf, 2008. Accessed October 2010.

[78] N. S. Networks. Reducing the impact of smartphone generated signaling traffic while increasing the battery life of the phone through the use of network optimization techniques. www.nokiasiemensnetworks.com/sites/default/files/document/Signals_Research_-_Smartphones_and_a_3G_Network_May_2010.pdf, 2010. Accessed March 2011.

[79] G. Newspaper. Half world's population 'will have mobile phone by end of year'. http://www.guardian.co.uk/technology/2008/sep/26/mobilephones.unitednations, 2008. Accessed March 2010.

[80] J. Nielson. One billion internet users. http://www.useit.com/alertbox/internet_growth.html, Dec. 2005. Accessed March 2010.

[81] J. Noyranen and J. Nurminen. Energy-consumption in mobile peer-to-peer quantitative results from file sharing. In *5th IEEE Consumer Communications & Networking Conference (CCNC '08)*, 2008.

[82] J. Nurminen and I. Kelenyi. Energy aspects of peer cooperation - measurements with a mobile DHT system. In *IEEE ICC Communications Workshops*, 2008.

[83] J. Nurminen and I. Kelenyi. Optimizing energy consumption of mobile nodes in heterogeneous kademlia-based distributed hash tables. In *2nd Next Generation Mobile Applications, Services, and Technologies (NGMAST '08)*, 2008.

[84] J. K. Nurminen, B. Bakos, and L. Farkas. P2P applications on smart phones using cellular communications. In *IEEE Wireless Communications and Networking Conference (WCNC '06)*, 2006.

[85] E. of Scotland KTP Centre. Knowledge transfer partnerships. http://www.ktpcentre.com/, 2009. Accessed November 2009.

[86] H. U. of Technology. Laboratory works in radio communications: GSM transceiver measurements. http://www.comlab.hut.fi/opetus/260/1v153.pdf, 2005. Accessed May 2011.

[87] G. P. Office. Press release: The GSMA reports record attendance at mobile world congress 2011. http://www.gsmworld.com/newsroom/press-releases/2011/6056.htm, 2011. Accessed June 2011.

[88] M. Ould-Khaoua and A. Al-Maashri. Performance analysis of MANET routing protocols in the presence of self-similar traffic. In *31st IEEE Conference on Local Computer Networks*, 2006.

[89] P2PON. Top 10 most popular P2P file sharing clients of 2010/2011. http://www.p2pon.com/top-10-most-popular-p2p-file-sharing-programs-in-2009-at-your-request/, 2009. Accessed March 2011.

[90] M. R. Pakravan, K. Ronasi, and M. H. Firooz. An enhanced random walk method for content locating in P2P networks. In *Distributed Computing Systems Workshops*, 2007.

[91] M. Patel. Driving high performance through mobile computing: Eight trends to watch, 2008. Presentation May 2008.

[92] V. S. Perreault. NAT and firewall traversal with STUN/TURN/ICE. http://www.viagenie.ca/publications/2008-09-24-astricon-stun-turn-ice.pdf, Sept. 2008. Presentation AstriCon 2008 Accessed May 2010.

[93] C. Pettey and H. Stevens. Gartner highlights key predictions for IT organizations and users in 2010 and beyond. http://www.gartner.com/it/page.jsp?id=1278413, 2010. Accessed May 2010.

[94] W. Pugh. Skip lists: A probablistic alternative to balanced trees. In *Communications of the ACM Vol. 33, No. 6*, 1990.

[95] L. Pupillo and E. Noam. *Peer to Peer Video. The Economics, Policy and Culture of Today's New Mass Medium*. Springer Science and Business Media, 2008.

[96] F. Reichert and F. H. P. Fitzek. *External Energy Consumption Measurements on Mobile Phones*. Springer - Mobile Phone Programming and its Application to Wireless Networking, 2007.

[97] R. Rejaie and D. Stutzbach. Technical report CIS-TR-2005-03 university of oregon: Characterizing churn in peer-to-peer networks. http://www.sis.pitt.edu/~oherrera/Research/PDFs/tr05-03.pdf, 2005. Accessed May 2011.

[98] A. Richa, G. Plaxton, and R. Rajaraman. Accessing nearby copies of replicated objects. In *9th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '97)*, 1997.

[99] S. Rodhetbhai, S. Naicken, A. Basu, and B. Livingston. A survey of peer-to-peer network simulators. In *7th Annual Postgraduate Symposium (PGNet '06) Liverpool John Moores University*, 2006.

[100] E. Rodriguez. Basic networking topology. http://www.skullbox.net/ntoplogy.php, 2010. Accessed November 2010.

[101] J. Rosenburg. Interactive connectivity establishment. http://www.isoc.org/tools/blogs/ietfjournal/?p=117, 2006. Accessed May 2010.

[102] J. Rosenburg. Interactive connectivity establishment (ICE) tutorial. http://www.jdrosen.net/papers/ice-basic-tutorial.pdf, 2006. Accessed May 2010.

[103] K. Ross and J. Kurose. *Computer Networking: A Top Down Approach Featuring the Internet*. Pearson Addison Wesley, 2005.

[104] K. W. Ross and J. Buford. P2P overlay design overview. In *IETF P2P-SIP*, 2005.

[105] S. Ross. Ofcom's call to arms over mobile phone 'not spots'. http://news.scotsman.com/mobilephonemasts/Ofcom39s-call-to-arms-over.5442459.jp, 2009. Accessed March 2010.

[106] A. Rowstron, M. Gerla, and C. Lindemann. P2P MANET's - new research issues. In *Dagstuhl Seminar Proceedings. Perspectives Workshop: Peer-to-Peer Mobile Ad Hoc Networks - New Research Issues*, 2005.

[107] D. Rubenstein and K. W. Ross. P2P systems. http://cis.poly.edu/~ross/tutorials/P2PtutorialInfocom.pdf, May 2004. Accessed March 2010.

[108] A. Saberi, C. Gkantsidis, and M. Mihail. Random walks in peer-to-peer networks. In *IEEE InfoCom Conference Proceedings*, 2004.

[109] A. E. Saddik and A. Roczniak. *Improving Robustness of P2P Applications in Mobile Environments*. Springer - Peer to Peer Networking and Applications, 2009.

[110] E. Saxe. Power efficient software systems. http://queue.acm.org/detail.cfm?id=1698225, 2010. Accessed January 2011.

[111] C. Sekaran and K. S. Thampi. Review of replication schemes for unstructured P2P networks. In *IEEE International Advance Computing Conference (IACC '09)*, 2009.

[112] K. Shen and M. Zhong. Popularity biased random walks for peer to peer searches under the square root principle. In *5th International Workshop on P2P Systems*, 2006.

[113] K. Shen and M. Zhong. Random walk based node sampling in self organizing networks. In *Operating Systems Review Vol.40 ACM SIGOPS*, 2006.

[114] S. Shenker and E. Cohen. Replication strategies in unstructured peer to peer networks. In *Applications, Technologies, Architectures and Protocols for Computer Communications Conference (ACM SIGCOMM '02)*, 2002.

[115] S. Shenker, S. Ratnasamy, P. Francis, M. Handley, and R. Karp. A scalable content-addressable network. In *Special Interest Group on Data Communication Conference (SIGCOMM '01)*, 2001.

[116] J. Sherwood. UK WAP usage doubles in 12 months. `http://www.builderau.com.au/news/soa/UK-WAP-usage-doubles-in-12-months/0,339028227,339130400,00.htm`. Accessed April 2010.

[117] T. Smith. World smartphone sales: Apple closes on RIM ... and becomes US leading phone maker. `http://www.reghardware.com/2010/04/30/q1_2010_smartphone_shipments/`, 2010. Accessed March 2011.

[118] D. Spinellis, V. Vlachos, and S. Androutsellis-Theotokis. Security applications of peer-to-peer network. *Computer Networks*, 45:195–205, 2004.

[119] M. V. Steen and A. Tanenbaum. *Distributed Systems - Principles and Paradigms*. Prentice Hall, 2002.

[120] I. Stoica, H. Balakrishnan, M. Kaashoek, D. Karger, and R. Morris. Looking up data in P2P systems. In *Communications of the ACM Vol. 46, No. 2*, 2003.

[121] TechAutos. Making sense of smartphone processors: The mobile CPU/GPU guide. `http://www.techautos.com/2010/03/14/smartphone-processor-guide/`, 2010. Accessed June 2010.

[122] S. Technologies. About skype VOIP. `http://about.skype.com`. Accessed April 2010.

[123] S. Technologies. Skype VOIP. `http://www.skype.com`. Accessed March 2010.

[124] B. Technology. European mobile data prices set. `http://news.bbc.co.uk/1/hi/technology/7965233.stm`, 2009. Accessed June 2010.

[125] R. Technology. Wireshark protocol analyzer. `http://www.wireshark.org/`. Accessed October 2010.

[126] G. Templeton and J. Bailes. Managing P2P security. *Communications of the ACM - End-user Development: Tools That Empower Users to Create Their Own Software Solutions*, 47, 2004.

[127] P. Thivent. Power efficient software coding. `http://stackoverflow.com/questions/61882/power-efficient-software-coding`, 2008. Accessed January 2011.

[128] M. Tiwari. Nokias e-series: The smartphones. `http://informationmadness.com/cms/technology/tech-tips/2436-nokias-e-series-the-smartphones.html`, 2010. Accessed June 2010.

[129] TNS. TNS mobile life research. `http://discovermobilelife.com/`, 2011. Accessed May 2011.

[130] K. Tutschku, L. Li, and J. Cao. *Editorial: Special Issue on Mobile P2P Networking and Computing*. Springer - Peer to Peer Networking and Applications, 2009.

[131] A. Venkataramani, N. Balasubramanian, and A. Balasubramanian. Energy consumption in mobile phones: A measurement study and implications for network applications. In *9th ACM SIGCOMM conference on Internet Measurement*, 2009.

[132] D. Wallach. A survey of peer-to-peer security issues. In *International Symposium on Software Security*, 2002.

[133] C. Wang and B. Li. Peer to peer overlay networks: A survey. Technical Report, Apr. 2003. Accessed December 2009.

[134] Y. Wang, F. Maker, and E. Jung. Presentation - university of california, davis: Mobile energy consumption. http://www.slideshare.net/marakana/learn-about-energy-consumption-and-battery-life-on-android-devices, 2010. Accessed January 2011.

[135] K. Wehrle and R. Steinmetz. *State of the Art Survey: Peer to Peer Systems and Applications*. Springer Verlag, 2005.

[136] T. Wenger. Skip list: A probabilistic alternative to balanced trees. http://iamwww.unibe.ch/~wenger/DA/SkipList/, 1997. Accessed December 2009.

[137] Wikipedia. Wikipedia entry: ARM 11 processors. http://en.wikipedia.org/wiki/ARM11. Accessed May 2011.

[138] Wikipedia. Wikipedia entry: ARM family processors. http://en.wikipedia.org/wiki/ARM_processor. Accessed May 2011.

[139] Wikipedia. Wikipedia entry: Bittorrent. http://en.wikipedia.org/wiki/BitTorrent_%28protocol%29. Accessed March 2011.

[140] Wikipedia. Wikipedia entry: Carbide.c++. http://en.wikipedia.org/wiki/Carbide.c%2B%2B. Accessed March 2011.

[141] Wikipedia. Wikipedia entry: edonkey network. http://en.wikipedia.org/wiki/EDonkey_network. Accessed March 2011.

[142] Wikipedia. Wikipedia entry: Fasttrack. http://en.wikipedia.org/wiki/FastTrack. Accessed March 2010.

[143] Wikipedia. Wikipedia entry: Gnutella. http://en.wikipedia.org/wiki/Gnutella. Accessed March 2011.

[144] Wikipedia. Wikipedia entry: Grokster. http://en.wikipedia.org/wiki/Grokster. Accessed March 2010.

[145] Wikipedia. Wikipedia entry: ICE. http://en.wikipedia.org/wiki/Interactive_Connectivity_Establishment. Accessed May 2010.

[146] Wikipedia. Wikipedia entry: Internet protocol version 6 (IPv6). http://en.wikipedia.org/wiki/IPv6. Accessed May 2011.

[147] Wikipedia. Wikipedia entry: IPv4 exhaustion. http://en.wikipedia.org/wiki/IPv4_address_exhaustion. Accessed May 2010.

[148] Wikipedia. Wikipedia entry: Kademlia. http://en.wikipedia.org/wiki/Kademlia. Accessed March 2010.

[149] Wikipedia. Wikipedia entry: Kazaa. http://en.wikipedia.org/wiki/Kazaa. Accessed March 2010.

[150] Wikipedia. Wikipedia entry: Mobile open source health care software. http://en.wikipedia.org/wiki/List_of_open_source_healthcare_software#Mobile_.2F_Handheld_Devices. Accessed September 2010.

[151] Wikipedia. Wikipedia entry: Network address translator. http://en.wikipedia.org/wiki/Network_address_translation. Accessed May 2010.

[152] Wikipedia. Wikipedia entry: Nokia 6120 classic. http://en.wikipedia.org/wiki/Nokia_6120_classic. Accessed May 2011.

[153] Wikipedia. Wikipedia entry: Nokia E52. http://en.wikipedia.org/wiki/Nokia_E52. Accessed May 2011.

[154] Wikipedia. Wikipedia entry: Nokia E75. http://en.wikipedia.org/wiki/Nokia_E75. Accessed May 2011.

[155] Wikipedia. Wikipedia entry: Nokia N79. http://en.wikipedia.org/wiki/Nokia_N79. Accessed May 2011.

[156] Wikipedia. Wikipedia entry: Nokia N95-8GB. http://en.wikipedia.org/wiki/Nokia_N95. Accessed May 2011.

[157] Wikipedia. Wikipedia entry: PowerTOP. http://en.wikipedia.org/wiki/PowerTOP. Accessed May 2011.

[158] Wikipedia. Wikipedia entry: S60 UI. http://en.wikipedia.org/wiki/S60_(software_platform). Accessed June 2010.

[159] Wikipedia. Wikipedia entry: Skip list. http://en.wikipedia.org/wiki/Skip_list. Accessed September 2010.

[160] Wikipedia. Wikipedia entry: STUN. http://en.wikipedia.org/wiki/Session_Traversal_Utilities_for_NAT. Accessed May 2010.

[161] Wikipedia. Wikipedia entry: Symbian OS. http://en.wikipedia.org/wiki/Symbian_OS. Accessed June 2010.

[162] Wikipedia. Wikipedia entry: TURN. http://en.wikipedia.org/wiki/Traversal_Using_Relay_NAT. Accessed May 2010.

[163] D. Wing. Overview of ICE. http://www.interop.com/lasvegas/2006/presentations/downloads/session-border-controllers-d-wing.pdf, 2006. Presentation InterOp 2006 Accessed February 2010.

[164] A. Wolfe. Is the smartphone your next computer? http://www.informationweek.com/news/personal_tech/smartphones/showArticle.jhtml?articleID=210605369, 2006. Accessed March 2010.

[165] X. Wu, K. Dhara, Y. Guo, and M. Kolberg. *Overview of Structured Peer to Peer Overlay Algorithms*. Springer Science and Business Media, 2010.

[166] M. Yarlagadda. Video - VOIP: Ready for primetime. http://www.youtube.com/watch?v=9MWYw0fltr0, 2006. Accessed May 2010.

[167] J. Yi, N. Saxena, and G. Tsudik. Admission control in peer-to-peer: Design and performance evaluation. In *1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2004.

[168] J. You, Z. Jia, R. Rao, and M. Li. *Random Walk Spread and Search in Unstructured P2P*, volume 3320/2005 of *Parallel and Distributed Computing: Applications and Technologies*. Springer Berlin / Heidelberg, 2005.

[169] H. Yu and X. Shen. *Editorial: Peer-to-Peer Networking and Applications (Vol 2. Iss 2.)*. Springer - Peer to Peer Networking and Applications, 2009.

[170] B. Zhao, S. Gurun, and P. Nagpurkar. Energy consumption and conservation in mobile peer-to-peer systems. In *First International Workshop on Decentralized Resource Sharing in Mobile Computing and Networking (MobiShare '06)*, 2006.

[171] X. Zhou, L. Chen, B. Cui, H. T. Shen, and W. Lu. Efficient information retrieval in mobile peer-to-peer networks. In *18th ACM Conference on Information and Knowledge Management (CIKM '09)*, 2009.